

Thema

On-the-Fly Recommendation Retrieval from Linked Open Data Repositories

Dissertation

zur Erlangung des akademischen Grades
doctor rerum politicarum
(Dr. rer. pol.)

vorgelegt dem
Rat der Wirtschaftswissenschaftlichen Fakultät
der Friedrich-Schiller-Universität Jena

am 23.05.2018

von: Lisa Wenige

geboren am: 15. November 1986 in: Leipzig

Gutachter:

1. Prof. Dr. Johannes Ruhland, Lehrstuhl für Wirtschaftsinformatik, FSU Jena
2. Prof. Dr. Nils Boysen, Lehrstuhl für Allgemeine Betriebswirtschaftslehre und Operations Management, FSU Jena

Datum der Verteidigung: 25.09.2018

Eidesstattliche Erklärung

Hiermit erkläre ich,

- dass mir die geltende Promotionsordnung der Wirtschaftswissenschaftlichen Fakultät der Friedrich-Schiller-Universität Jena bekannt ist,
- dass ich die Dissertation selbst angefertigt, keine Textabschnitte eines Dritten oder eigene Prüfungsarbeiten ohne Kennzeichnung übernommen und alle von mir benutzten Hilfsmittel, persönlichen Mitteilungen und Quellen in meiner Arbeit angegeben habe,
- dass die Hilfe eines Promotionsberaters nicht in Anspruch genommen wurde und dass Dritte weder unmittelbar noch mittelbar geldwerte Leistungen von mir für Arbeiten erhalten haben, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen,
- dass ich die Dissertation noch nicht als Prüfungsarbeit für eine staatliche oder andere wissenschaftliche Prüfung eingereicht habe,
- dass ich weder die gleiche, eine in wesentlichen Teilen ähnliche oder eine andere Abhandlung bei einer anderen Hochschule bzw. anderen Fakultät als Dissertation eingereicht habe,
- dass mich bei der Auswahl und Auswertung des Materials sowie bei der Herstellung des Manuskriptes mein Promotionsbetreuer Prof. Dr. Johannes Ruhland unterstützt hat.

.....
Ort, Datum

.....
Lisa Wenige

Für meine Eltern, meinen Bruder und meinen Sohn

Danksagung

Am Gelingen einer Promotion haben immer viele Personen einen Anteil. Zuallererst müssen natürlich die finanziellen und organisatorischen Rahmenbedingungen stimmen. Deshalb möchte ich meinen besonderen Dank an Prof. Ruhland aussprechen, der es mir durch ein flexibles Arbeitszeitmodell ermöglicht hat, die Dissertation neben der Lehrstuhltätigkeit und meiner familiären Situation fertigzustellen. Ohne diese Unterstützung und das Vertrauen wäre es nicht möglich gewesen, das Promotionsvorhaben in einem vergleichbaren Zeitrahmen zu beenden. Außerdem möchte ich Prof. Ruhland für die vielen fachlichen Ratschläge und wertvollen Hinweise danken, die geholfen haben, die Arbeit zu verbessern.

Großer Dank geht auch an meine lieben Zimmerkollegen am Lehrstuhl, die mich über die Jahre begleitet haben. Ihr habt geholfen, so manchen stressigen Tag oder „Forschungshänger“ durch einen Witz oder ein kurzes Pausengespräch wieder zu erhellen. Danke Uli, Martin S., Hartmut und Geraldine! Aber auch die anderen Kollegen am Lehrstuhl waren stets eine große Hilfe bei fachlichen Fragen und haben mit ihrer Freundlichkeit zu einer guten Arbeitsatmosphäre beigetragen. Ein herzlicher Dank geht deshalb auch an Birgit, Christian, Marek, Sandra und Sven. Außerdem kann ich Sven, Sandra und Martin S. gar nicht genug für die schnelle und zuverlässige Unterstützung in der Endphase der Dissertation danken. Ihr habt mir durch eure Anregungen sehr wertvolle Denkanstöße gegeben. Große Dankbarkeit gebührt auch Verena für die hilfreichen Hinweise zur Verbesserung des Manuskripts. Aber du hast mir nicht nur in fachlicher Hinsicht zur Seite gestanden, Verena, sondern warst schon von der ersten Stunde an eine treue Begleiterin und Vertraute in den Lehrstuhljahren. Du hast mir unzählige Male geholfen, nicht nur durch dein stets offenes Ohr, sondern auch in ganz praktischer Hinsicht, wenn Not am Mann war. Lieben Dank für deine Unterstützung!

Großer Dank geht auch an Josh und Rike für die schnelle und zuverlässige sprachliche Korrektur des Manuskripts. Ihr habt mir als Nicht-Muttersprachlerin dabei geholfen, grobe Schnitzer im Englischen zu vermeiden. Rike, dir möchte ich außerdem dafür danken, dass du für mich da warst und in den letzten Jahren immer wieder aufmunternde Worte für mich gefunden hast. Auch andere Freunde haben mir oft ganz viel Kraft und Freude gegeben, um nicht an dem Berg an Arbeit zu verzagen. Danke Knopf und Thea für eure Freundschaft und den Halt, den ihr mir gebt. Lieben Dank auch dir, Anne, für die stets guten Gespräche, deine Wertschätzung und die gemeinsam verbrachten Stunden am Wochenende.

Außerdem möchte ich Jan danken, der mich zuverlässig und schnell bei der Fertigstellung der Formatierung unterstützt hat. Ohne deine Zauberkünste hätte ich noch zusätzliche Nachtschichten einlegen müssen. Mein großer Dank gebührt auch Tino und Catharina, die zuhause präsent

waren, wenn ich nicht da war. Ohne eure liebevolle Betreuung wäre vieles nicht möglich gewesen. Vielen Dank!

Schließlich möchte ich noch meinen Eltern und meinem Bruder meinen unendlichen Dank aussprechen. Ihr habt mir auf vielfältige Weise unzählige Male geholfen. Sei es in der Endphase der Promotion durch die Schaffung von zeitlichen Freiräumen, durch tatkräftige Unterstützung in der praktischen Alltagsorganisation oder durch die zahlreichen guten Gespräche. Ich kann euch gar nicht genug danken für alles, was ihr für mich getan habt. Ohne eure Fürsorge und Hilfe stünde ich jetzt nicht an diesem Punkt. Tausend Dank!

Kurzfassung

Die vorliegende Dissertation beschäftigt sich mit der Frage, inwieweit die öffentlich zugänglichen Wissensressourcen der Linked Open Data (LOD) Cloud besser für Empfehlungsaufgaben genutzt werden können. Zwar setzen einige Empfehlungssysteme bereits RDF-Daten ein, um die lokale Datenbasis anzureichern, allerdings nutzen diese noch nicht alle Möglichkeiten, die das Datenweb aktuell zu bieten hat.

Ausgehend von den zentralen Stärken typischer LOD-Datensammlungen (Aktualität, Vollständigkeit, sowie rechtliche und technische Offenheit der Daten) und der besonderen Eignung des Technologie-Stacks der LOD Cloud für Empfehlungsaufgaben (Ausdrucksmächtigkeit der graphbasierten Datenstruktur) auf der einen Seite, sowie den Herausforderungen der Verarbeitung verlinkter RDF-Daten (Quantität, Heterogenität und Verteilung der Daten) auf der anderen Seite, wurde in der Arbeit eine Anforderungsspezifikation für ein LOD-Empfehlungssystem erstellt. Mit dieser Spezifikation sollen die Eigenschaften des öffentlichen Datenwebs adäquat durch Systemfunktionalitäten beschrieben werden. Auf Basis der Literatur in diesem Themenfeld und den geforderten Komponenten der Anforderungsspezifikation wurde das Empfehlungssystem SKOSRecommender (SKOSRec) entwickelt. Zentraler Bestandteil dieses Systems ist, neben der Nutzung von SKOS-Annotationen zur Identifizierung ähnlicher Objekte, eine graphbasierte Abfragesprache (SKOSRec query language), die sich SPARQL-ähnlicher Konstrukte bedient, um Daten aus LOD-Repositoryn zu extrahieren. Durch die Verwendung des SKOS-Standards können zudem zusätzliche ähnliche Deskriptoren in das Retrieval einbezogen werden. In diesem Zusammenhang wurde ein neuartiges Verfahren der Anfrageerweiterung mit Namen *Flexible Similarity Detection* entwickelt. Zusätzlich zur Möglichkeit der Nutzung ähnlicher Konzepte, hat der Einsatz von SKOS-Vokabularen auch den Vorteil, dass deklarierte Mapping-Beziehungen zwischen Deskriptoren verschiedener Vokabulare für repositoryübergreifende Anfragen verwendet werden können. Diese Daten werden im SKOSRec-System für die Erstellung sogenannter *Cross-Repository Recommendations* eingesetzt.

Ein weiterer integraler Bestandteil des im Rahmen der Dissertation entwickelten Empfehlungssystems ist das Verfahren der Adhoc-Erstellung von Vorschlägen (*Fast On-the-fly Retrieval*) auf Basis von Nutzervorlieben. Mit diesem Verfahren können durch effiziente Abfragen von SPARQL-APIs zur Laufzeit LOD-Empfehlungen generiert werden. Auf diese Weise ist es möglich, SPARQL-ähnliche Abfragemuster mit suchbasierten Verfahren flexibel zu kombinieren. Die im Zuge der Arbeit entwickelte Retrievalsprache hilft dabei, die individuellen Präferenzen eines Nutzers entweder in Form von konkreten Vorlieben für bestimmte Objekte oder mithilfe abfragebasierter Elemente abzubilden. Außerdem können z.B. Empfehlungsergebnisse als Teil einer graphbasierten Unterabfrage genutzt werden. Dies ermöglicht neuartige Anfragetypen und fortgeschrittene Empfehlungsstrategien. Mittels graphbasierter Abfragemuster lassen sich beispielsweise ausdrucks mächtige Filterbedingungen für Ergebnislisten formulieren (*Expressive Constraint-based Queries*). Des Weiteren ist es mit einer einzigen Anfrage möglich, zunächst

automatisch ein Nutzerprofil zu erstellen, dass dann für die Ermittlung von passenden Empfehlungen verwendet wird (*Preference Querying*). Außerdem lässt es die SKOSRec-Sprache zu, die verschiedenen Abfragemuster so zu kombinieren, dass semantisch ausdrucksstarke Empfehlungsanfragen (*Advanced Queries*) an LOD-Repositories gestellt werden können. Beispiele für solche Anfragen sind das *Rollup*-Retrievalmuster oder *Cross-Domain-Abfragen*.

Die entwickelten neuartigen Empfehlungsstrategien wurden in einer Reihe von Offline- und Onlineexperimenten vor dem Hintergrund passender LOD-basierter Empfehlungsszenarien (Reiseempfehlungen, Multimediaempfehlungen, Empfehlungen im Bereich der wissenschaftlichen Publikationssuche) evaluiert. Dabei zeigte sich, dass die entwickelten Ansätze (z.B. *Flexible Similarity Detection*, *Expressive Constraint-based* und *Cross-Domain Queries*) tatsächlich die Empfehlungsqualität herkömmlicher content-basierter Verfahren verbessern konnten. Die Effekte kamen v.a. in den Performanzdimensionen *Recall* (dt. Trefferquote), der Diversität und dem Neuigkeitsgehalt von Empfehlungen zum Tragen. Für andere Verfahren, wie z.B. für das *Rollup*-Anfragemuster, konnten zwar keine signifikanten Verbesserungen erreicht werden, der Ansatz erzielte aber zumindest gleichwertige Qualitätswerte, sodass er eine geeignete alternative Retrievalstrategie darstellt.

Mit den neuartigen Methoden der Empfehlungsgenerierung und den überwiegend positiven Evaluationsergebnissen leistet die vorliegende Arbeit einen Beitrag zur Weiterentwicklung personalisierter Filtertechniken, die sowohl für das semantische Retrieval in LOD-Repositories, als auch für klassische Empfehlungsaufgaben geeignet sind.

Abstract

This thesis investigates the potential of Linked Open Data (LOD) for recommendation tasks. Whereas some existing recommender systems (RS) already utilize RDF data to enhance the local database, they do not yet take full advantage of the potential of the LOD cloud.

The work describes the strengths of LOD repositories (timeliness, comprehensiveness, legal and technical openness), the suitability of the LOD technology stack (e.g., expressive data models) and the challenges of RDF processing (data quantity, data quality, and data distribution) for recommendation tasks. A requirements specification for a LOD-enabled RS was defined based upon these features. The specification addresses the characteristics of the openly accessible data web to utilize its full potential for personalized retrieval.

Based on a literature survey as well as on the statements of the requirements specification, a recommendation engine, called SKOSRecommender (SKOSRec), was developed. Aside from the usage of SKOS annotations, which are processed to determine similar items, a graph-based query language is also part of the system's components. The SKOSRec query language utilizes SPARQL-like retrieval patterns to extract data from LOD repositories. The SKOS standard facilitates usage of related descriptors to broaden retrieval. In this context, a novel concept ex-

pansion method (i.e., *flexible similarity detection*) is introduced. In addition to the provision of similar concepts, SKOS vocabularies contain mapping relations to descriptors of other vocabularies, which can facilitate *cross-repository recommendations*.

Another important feature is the engine's capability to generate ad-hoc suggestions based on item-specific user preferences (i.e., *fast on-the-fly retrieval*). With this method, the engine calculates recommendations at runtime. Thus, it is possible to combine SPARQL-like query patterns with search-based procedures flexibly. In this context, the SKOSRec query language enables representation of individual user preferences either through explicit statements or by using query-based elements. Additionally, the language facilitates application of recommendation results as part of a graph-based subquery. It enables novel query types as well as advanced retrieval approaches. For instance, graph-based query patterns can be used to formulate powerful filter conditions for result lists (*expressive constraint-based queries*). It is also possible to generate a user profile with the help of a SPARQL-like request (*preference querying*). Besides, the SKOSRec query language allows combinations of graph- and search-based query patterns (i.e., *advanced recommendation requests*). Examples of such requests are *rollup retrieval patterns* or *cross-domain queries*.

The author evaluated the novel approaches in a series of offline and online experiments in the context of suitable scenarios for LOD-enabled recommendations, namely travel RS, multimedia RS and scientific publication retrieval. The results show that some of the developed methods (e.g., *flexible similarity detection*, *expressive constraint-based queries*, *cross-domain queries*) improve the quality of conventional content-based recommendation methods in certain application scenarios. Effects predominantly occurred in the performance dimensions of *recall*, *novelty*, and *diversity*. Other retrieval approaches (e.g., *rollup queries*), while not achieving significantly increased assessments, had similar scores. Hence, they are still a viable alternative retrieval strategy.

With the novel recommendation methods and the predominantly positive evaluation results, the present work contributes to the advancement of personalized search techniques, which can be applied for semantic retrieval in LOD repositories as well as for classical recommendation tasks.

Contents

List of Tables	V
List of Figures	IX
List of Abbreviations	XIII
List of Symbols	XVII
List of Equations	XXIII
List of Listings	XXV
List of Definitions	XXVII
1 Introduction	1
1.1 Motivation	1
1.2 Research Objectives	5
1.3 Thesis Outline	6
2 Recommender Systems	9
2.1 General Characteristics of Recommender Systems	9
2.2 Recommendation Algorithms	12
2.2.1 Collaborative Filtering Recommender Systems	12
2.2.2 Content-based Recommender Systems	14
2.2.3 Knowledge-based Recommender Systems	16
2.2.4 Hybrid Recommendation Approaches	18
2.3 Limitations of Recommender Systems	19
3 Linked Open Data for Recommendation Tasks	23
3.1 Emergence and State of the Linked Open Data Cloud	23
3.1.1 Semantic Web	23
3.1.2 Linked Open Data	24
3.1.3 RDF	26
3.1.4 SPARQL	29
3.2 Opportunities of Linked Open Data for Recommendation Tasks	32

3.2.1	Openness	32
3.2.2	Comprehensiveness	33
3.2.3	Timeliness	35
3.2.4	Data Expressiveness & Inference	36
3.3	Challenges of Linked Open Data for Recommendation Tasks	39
3.3.1	Data Quantity	39
3.3.2	Data Heterogeneity	40
3.3.3	Data Distribution	41
4	Requirements Analysis	45
4.1	Characteristics of a Requirements Analysis for a Linked Open Data-enabled Recommender System	45
4.2	Functional Requirements	46
4.2.1	Requirements to Similarity Detection	46
4.2.2	Requirements to Linked Open Data Integration	48
4.2.3	Requirements to Virtual Data Integration	49
4.3	Performance Requirements	50
4.4	Non-functional Requirements	51
5	Related Work	53
5.1	Non-Linked Data Systems	53
5.1.1	Search Systems	53
5.1.2	Query-based Recommender Systems	56
5.2	Linked Data-enabled Systems	59
5.2.1	Linked Data Search Systems	59
5.2.2	Linked Data Recommender Systems	61
5.2.3	Query-based Linked Data Recommender and Search Systems	64
5.3	Summary & Research Agenda	65
6	The SKOS Recommender	67
6.1	Simple Knowledge Organization Systems	67
6.2	Usage Scenarios	70
6.2.1	Scenario Selection	70
6.2.2	LOD Repositories	72
6.3	On-the-Fly Recommendations	75
6.4	Fast On-the-Fly Recommendations	82
6.5	Flexible Similarity Detection	84
6.5.1	Concept Expansion	84
6.5.2	Concept-to-Concept Similarity	86
6.5.3	Item Similarities with Concept Expansion	88

6.6	Constraint-based Recommendations	93
6.6.1	Prefiltering	93
6.6.2	Preference Querying	96
6.6.3	Postfiltering	98
6.7	Cross-Repository Recommendations	101
6.8	The SKOS Recommender Query Language	104
6.8.1	Syntax	104
6.8.2	Examples	107
6.8.2.1	On-the-Fly Recommendations	107
6.8.2.2	Recommendations from Flexible Similarity Detection	108
6.8.2.3	Constraint-based Recommendations	109
6.8.2.4	Combinations	112
6.9	Implementation	118
7	Evaluation of the SKOS Recommender	125
7.1	Evaluation Methods	125
7.1.1	Prototypical Implementation	125
7.1.2	Offline Experiments	126
7.1.3	User experiments	127
7.1.4	Performance Dimensions	129
7.1.4.1	Accuracy	129
7.1.4.2	Novelty	131
7.1.4.3	Diversity	132
7.1.4.4	Perceived Usefulness	133
7.2	Evaluation of Performance Requirements	134
7.2.1	Methods for Performance Evaluation	134
7.2.2	Results of the Performance Evaluation	136
7.3	Evaluation of General Recommendation Quality	137
7.3.1	Methods for the Evaluation of General Recommendation Quality	137
7.3.2	Results of the Evaluation of General Recommendation Quality	146
7.4	Evaluation of Flexible Similarity Detection	154
7.4.1	Methods for the Evaluation of Flexible Similarity Detection	154
7.4.2	Results of the Evaluation of Flexible Similarity Detection	159
7.5	Evaluation of Constraint-based Recommendation Retrieval	163
7.5.1	Methods for the Evaluation of Constraint-based Recommendation Retrieval	163
7.5.2	Results of the Evaluation of Constraint-based Recommendation Retrieval	169
7.6	Evaluation of Advanced Query Patterns	172
7.6.1	Methods for the Evaluation of Advanced Query Patterns	172

7.6.2	Results of the Evaluation of Advanced Query Patterns	179
8	Conclusion	187
8.1	Summary	187
8.2	Discussion	192
A	Code Examples of the SKOS Recommender	195
A.1	Configuration and Query Input	195
A.2	SKOSRec Grammar	196
A.3	SKOSRec Query	199
A.4	SKOSRec Compiler	202
A.5	Similarity Calculation	204
A.6	Ranker	206
A.7	Optimizer	207
B	Supplementary Material of the Evaluation	209
B.1	Simulation - Flexible Similarity Detection	209
B.2	Webinterfaces of the Online Studies	211
B.2.1	Digital Library Experiment	211
B.2.2	Travel Experiment	219
B.2.3	Multimedia Experiments (Music Domain)	229
B.3	Correlation Matrices	242
	Bibliography	245

List of Tables

2.1	Example user-item rating matrix	12
2.2	Example item features of DBpedia movies [245]	15
2.3	Strengths and weaknesses of recommendation paradigms	21
3.1	Solution set of a simple SPARQL query	30
3.2	Solution set of a SPARQL query with multiple matchings	30
3.3	Overview of LOD domains in reference to [215]	34
5.1	Summary of IR approaches	56
5.2	Summary of query-based RS	59
5.3	Summary of Linked Data search engines	61
5.4	Summary of Linked Data recommender systems (LDRS)	63
5.5	Summary of query-based LDRS and search systems	65
6.1	Overview of the usage scenarios	72
6.2	Entity types in the usage scenarios	73
6.3	LOD repositories to be applied in the usage scenarios	75
6.4	Result set of Q1	108
6.5	Result set of Q2	109
6.6	Result set of Q3	109
6.7	Result set of Q4	110
6.8	Result set of Q5	110
6.9	Result set of Q6	111
6.10	Result set of Q7	112
6.11	Result set of Q8	112
6.12	Result set of Q9	114
6.13	Result set of Q10	114
6.14	Result set of Q11	116
6.15	Result set of Q12	116
6.16	Result set of Q13	117
6.17	Result set of Q14	118
6.18	<i>On-the-fly recommendation</i> workflow	121
6.19	<i>Constraint-based recommendation</i> workflow	123
6.20	<i>Cross-repository recommendation</i> workflow	124

7.1	Contingency table for relevance prediction	129
7.2	Overview of the LOD repositories applied in computational simulation runs . .	135
7.3	Results of the performance test (workloads)	136
7.4	Results of the performance test (execution times)	137
7.5	Details of the study series	139
7.6	Test cases in the web experiments	145
7.7	Cronbach's α values for the concept of <i>perceived usefulness</i>	149
7.8	Performance results of regular <i>on-the-fly retrieval</i> , part I	151
7.9	Performance results of regular <i>on-the-fly retrieval</i> , part II	152
7.10	Spearman's ρ for evaluation metrics correlation with <i>perceived usefulness</i> , TC1	152
7.11	Spearman's ρ for dependencies among evaluation metrics, TC1	153
7.12	Historical Dataset Features	155
7.13	Performance results of the simulation runs (multimedia domain)	161
7.14	Significant differences according to Friedman tests (multimedia domain)	162
7.15	Significant differences between the retrieval strategies as verified by Wilcoxon post-hoc tests (multimedia domain)	162
7.16	JI of recommendation lists for Tflex	162
7.17	Evaluation scores for <i>on-the-fly</i> (exact) vs. <i>flexible similarity detection</i> (DL domain)	163
7.18	SKOSRec filter options for <i>constraint-based recommendation</i> requests	166
7.19	Response rates for TC2	170
7.20	Performance results of <i>constraint-based recommendation</i> retrieval	171
7.21	JI of recommendation lists for TC2	172
7.22	<i>On-the-fly</i> vs. <i>advanced rollup query</i> patterns in the travel experiment	175
7.23	<i>On-the-fly</i> vs. <i>advanced rollup query</i> patterns in the multimedia experiments . .	176
7.24	Parameter for <i>advanced rollup query</i> patterns in the multimedia experiments . .	176
7.25	SPARQL vs. SKOSRec query patterns for <i>cross-domain recommendation</i> retrieval	178
7.26	Response Rates for TC3	179
7.27	Performance results of <i>on-the-fly</i> and <i>rollup recommendation</i> retrieval	180
7.28	JI of recommendation lists for TC3	181
7.29	Response rates for TC4	181
7.30	Performance results of SPARQL- and SKOSRec-based <i>cross-domain queries</i> .	183
7.31	JI of recommendation lists for TC4	184
7.32	Performance results of <i>on-the-fly</i> (regular) and SKOSRec <i>cross-domain queries</i>	184
8.1	Functional & Qualitative System Features	192
B.1	Spearman's ρ for evaluation metrics correlation with <i>perceived usefulness</i> , TC2	242
B.2	Spearman's ρ for dependencies among evaluation metrics, TC2	242

B.3	Spearman's ρ for evaluation metrics correlation with <i>perceived usefulness</i> , TC3	243
B.4	Spearman's ρ for dependencies among evaluation metrics, TC3	243
B.5	Spearman's ρ for evaluation metrics correlation with <i>perceived usefulness</i> , TC4	243
B.6	Spearman's ρ for dependencies among evaluation metrics, TC4	243

List of Figures

3.1	Semantic Web technology stack [136]	24
3.2	LOD cloud 2007	26
3.3	LOD cloud 2014	26
3.4	LOD cloud 2017	26
3.5	Example RDF graph	27
5.1	Online public access catalog (OPAC) of the Thuringian University and Regional Library (ThULB) Jena	53
6.1	Item features of an example LOD resource	68
6.2	Main elements of the SKOS data model [24]	69
6.3	<i>Fast on-the-fly recommendations</i>	85
6.4	Example subordinate/superior relations in the STW vocabulary	85
6.5	RDF representation of a similarity object of two STW concepts	89
6.6	Similarity detection with concept expansion	93
6.7	<i>On-the-fly vs. constraint-based retrieval</i> (travel domain)	96
6.8	<i>Constraint-based retrieval</i> with graph pattern matching (travel domain)	97
6.9	<i>Preference querying</i> (movie domain)	98
6.10	<i>Postfiltered recommendation</i> (aggregation-based)	102
6.11	<i>Cross-repository retrieval</i>	103
6.12	Architecture of the SKOSRec engine	119
6.13	Workflow of <i>on-the-fly retrieval</i>	121
6.14	Workflow of <i>constraint-based retrieval</i>	122
6.15	Workflow of <i>cross-repository retrieval</i>	123
7.1	Evaluation framework by Pu et al. [193]	135
7.2	Exec. time in the travel domain	138
7.3	Exec. time in the DL domain	138
7.4	Exec. time in the movie domain	138
7.5	Exec. time in the book domain	138
7.6	Exec. time in the music domain	138
7.7	Music - Demographics section (page 2)	140
7.8	Music - Consent form (page 1)	141
7.9	Error page	142

7.10	Music - User profile generation, TC1 (page 2)	142
7.11	Music - Results part I, TC1 (page 4)	144
7.12	Music - Results part II, TC1 (page 4)	145
7.13	Domain-wise gender distribution	146
7.14	Domain-wise age distribution	147
7.15	User perceptions of the search in their domains of interest	147
7.16	Distribution of session durations in the different domains	148
7.17	User agreement to positive statements on the <i>perceived usefulness</i> of <i>on-the-fly recommendations</i>	150
7.18	Results of the experiments on parameter configuration (γ parameter)	156
7.19	Results of the experiments on parameter configuration (distance parameter)	157
7.20	DL - Results, TFlex (page 6)	158
7.21	Results of offline simulation runs regarding <i>accuracy</i>	160
7.22	Results of offline simulation runs regarding <i>novelty</i> and <i>diversity</i>	161
7.23	Music - User profile generation, TC2 (page 5)	164
7.24	Domain-wise agreement ratios with the statement that the filter has improved the result list	169
7.25	Participants' overall satisfaction with <i>constraint-based recommendation</i> retrieval	170
7.26	Travel - User profile generation, TC3 (page 7)	173
7.27	Music - User profile generation, TC4 (page 9)	176
7.28	Participants' overall satisfaction with <i>on-the-fly</i> (regular) and <i>rollup requests</i>	179
7.29	Participants' overall satisfaction with <i>on-the-fly</i> (regular) and <i>cross-domain requests</i>	182
7.30	Mean recommendation list <i>sizes</i> of SPARQL (regular) and SKOSRec <i>cross-domain queries</i>	183
B.1	DL - Language selection (page 1)	211
B.2	DL - Demographics section (page 2)	211
B.3	DL - User profile generation, TC1 (page 3)	212
B.4	DL- Results part I, TC1 (page 4)	212
B.5	DL - Results part II, TC1 (page 4)	213
B.6	DL - User profile generation, TFlex (page 5)	213
B.7	DL - Results part I, TFlex (page 6)	214
B.8	DL - Results part II, TFlex (page 6)	215
B.9	DL - User profile generation, TC2 (page 7)	216
B.10	DL - Results, TC2 (page 8)	217
B.11	DL - Finish screen (page 9)	217
B.12	Travel - Consent form (page 1)	218
B.13	Travel - Demographics section (page 2)	219

B.14 Travel - User profile generation, TC1 (page 3)	220
B.15 Travel - Results part I, TC1 (page 4)	220
B.16 Travel - Results part II, TC1 (page 4)	221
B.17 Travel - User profile generation, TC2 (page 5)	222
B.18 Travel - Results part I, TC2 (page 6)	223
B.19 Travel - Results part II, TC2 (page 6)	224
B.20 Travel - Results part III, TC2 (page 6)	225
B.21 Travel - User profile generation, TC3 (page 7)	225
B.22 Travel - Results part I, TC3 (page 8)	226
B.23 Travel - Results part II, TC3 (page 8)	227
B.24 Travel - Results part III, TC3 (page 8)	228
B.25 Travel - Finish screen (page 9)	229
B.26 Music - Consent form (page 1)	229
B.27 Music - Demographics section (page 2)	230
B.28 Music - User profile generation, TC1 (page 3)	230
B.29 Music - Results part I, TC1 (page 4)	231
B.30 Music - Results part II, TC1 (page 4)	232
B.31 Music - User profile generation, TC2 (page 5)	233
B.32 Music - Results part I, TC2 (page 6)	234
B.33 Music - Results part II, TC2 (page 6)	235
B.34 Music - Results part III, TC2 (page 6)	236
B.35 Music - User profile generation, TC3 (page 7)	236
B.36 Music - Results part I, TC3 (page 8)	237
B.37 Music - Results part II, TC3 (page 8)	238
B.38 Music - Results part III, TC3 (page 8)	239
B.39 Music - User profile generation, TC4 (page 9)	239
B.40 Music - Results part I, TC4 (page 10)	240
B.41 Music - Results part II, TC4 (page 10)	241
B.42 Music - Finish screen (page 11)	242

List of Abbreviations

AGRIS	International System for Agricultural Science and Technology
AJAX	Asynchronous JavaScript and XML
ANOVA	Analysis of Variance
API	Application Programming Interface
AQE	Automatic Query Expansion
ARQ	Apache Jena SPARQL Query Processor
BNF	Backus Naur Form
CB	Content-based
CC	Creative Commons
CF	Collaborative Filtering
CP	Ceteris Paribus
CPU	Central Processing Unit
DAG	Directed Acyclic Graph
DCMI	Dublin Core Metadata Initiative
DL	Digital Library
DQP	Distributed Query Processing
EKP	Encyclopedic Knowledge Pattern
FAO	Food and Agriculture Organization of the United Nations
FDR	False Discovery Rate
FOAF	Friend of a Friend Ontology
GB	Gigabyte
GND	Integrated Authority Files (Gemeinsame Normdatei)
GO	Gene Ontology
HTTP	Hypertext Transfer Protocol
IC	Information Content
IEEE	Institute of Electrical and Electronics Engineers
IR	Information Retrieval
IRI	Internationalized Resource Identifier
JAVACC	Java Compiler Compiler
JI	Jaccard Index
KB	Knowledge-based
KOS	Knowledge Organization System

LSCH	Library of Congress Subject Headings
LDA	Latent Dirichlet Allocation
LDIF	Linked Data Integration Framework
LDRS	Linked Data Recommender Systems
LDS	Linked Data Semantic Distance
LOD	Linked Open Data
LSI	Latent Semantic Indexing
MAE	Mean Absolute Error
MESH	Medical Subject Headings
NLP	Natural Language Processing
NTO	Normalized Term Overlap
ODbL	Open Database License
ODC	Open Data Commons
OLAP	Online Analytical Processing
OPAC	Online Public Access Catalog
OWL	Web Ontology Language
PC	Personal Computer
POI	Point of Interest
RAM	Random-Access Memory
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
RFC	Request for Comments
RQL	RDF Query Language
RS	Recommender System
RSME	Root Mean Square Error
SKOS	Simple Knowledge Organization System
SKOSRec	SKOSRecommender
SPARQL	SPARQL Protocol And RDF Query Language
SQL	Structured Query Language
SRS	Software Requirements Specification
STW	Standard Thesaurus for Economics
SUMI	Software Usability Measurement Inventory
TAM	Technology Acceptance Model
TF-IDF	Term Frequency Inverse Document Frequency
TO	Term Overlap
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WWW	World Wide Web

ZBW

Leibniz Information Centre for Economics

List of Symbols

U	Set of users
u	A single user
R	Set of items
r	A single item / LOD resource
$Pref$	Rating matrix
$userSim(a, b)$	User-based similarity (CF)
$pref_{a,r}$	Preference/rating of user a for item r
\overline{pref}_a	Mean preference score of user a
$pred(a, r)$	Predicted preference score of user a for item r
Nb	User neighborhood
$itemSim(i, j)$	Item-based similarity (CF)
$ratedItems(u)$	Rated items of user u
K	Set of keywords
key	A single keyword
$w_{m,r}$	Weight of keyword m in d_r
\vec{d}_r	Weighted document vector of r
$freq(m, r)$	Frequency of keyword m in d_r
$TF(key_m, d_r)$	Term frequency of keyword key_m in d_r
$max_z freq(z, r)$	Maximum term frequency of a keyword in d_r
$freq(m)$	Number of occurrences of a keyword in a corpus
N	Number of items / participants

$IDF(key_m)$	Inverse document frequency of a keyword
$TFIDF(key_m, d_r)$	Term frequency - inverse document frequency of a keyword in d_r .
$docSim(d_i, d_j)$	Similarity of two documents representing items i and j
I	Set of IRIs
L	Set of literals
B	Set of blank nodes
p	Property
o	Object
D	RDF dataset
IL	Union of the sets of IRIs and literals
T	Set of RDF terms
t	A triple pattern
V	Set of variables
μ	Partial mapping function from V to T
$\mu(t)$	Mapping of a triple pattern
$[[t]]_D$	Evaluation of a triple pattern t over a dataset D
$var(t)$	Set of variables in a triple pattern
$dom(\mu)$	Set of those variables of a triple pattern that can be mapped to D
P	A graph pattern
Ω	Multiset of mappings according to μ
S	(Simple) Knowledge Organization System
C	Set of concepts
c	A single concept
st	SKOS annotation triple
AD	SKOS annotation dataset

$Annot(r)$	Set of SKOS annotations of resource r
P_r	Graph pattern to retrieve relevant resources and annotations for r
Ω_r	Solution mappings for P_r
$sim(r, q)$	Content-based similarity of two LOD resources r and q
C_{shared}	Set of shared concepts
$IC(C_{shared})$	Information content of the set of shared concepts
$freq(c)$	Concept frequency of c in the set of relevant resources
n	Maximum frequency among the concepts of relevant resources
Pr	User profile (Set of LOD resources)
$score(Pr, q)$	Recommendation score of a relevant resource q for a user profile Pr
k	Number of recommendations to be obtained
m	Number of shared annotations for <i>fast on-the-fly retrieval</i>
cut	Maximum number of shared annotations
$\Omega_{reduced}$	Reduced solution mappings of items
Ω_{cut}	Reduced solution mappings of relevant resources and annotations for <i>fast on-the-fly retrieval</i>
$CA_{v,z}$	Common ancestors of two concepts v and z
$\theta(c)$	Informativeness of a concept c in a taxonomy graph
$\Theta_{v,z}$	Informativeness values of common ancestors of two concepts v and z
$MICA_{v,z}$	Most informative common ancestor(s) of v and z
$\theta_{depth}(c)$	Informativeness based on concept depth
$Desc(c)$	Number of concepts (descendants) subsumed by c
$\theta_{desc}(c)$	Informativeness based on the number of descendants
$conceptSim(v, z)$	Concept-to-concept similarity of two concepts v and z
$p(v, z)$	Pondering function
γ	Pondering function parameter

$d(v, z)$	Path distance of v and z in the taxonomy graph
$conceptSim_{hyP}(v, z)$	Concept-to-concept similarity of two concepts v and z calculated with the hyP metric
ϵ	Concept-to-concept similarity threshold
$Prox(c, r)$	Proximate concepts of c for LOD resource r
$Prox(r)$	Union of all proximate concept sets of the SKOS annotations of r (Expanded concept set)
P_{exp}	Graph pattern to obtain items that are annotated with one or more concepts of the expanded concept set
Ω_{exp}	Solution mappings of relevant resources and annotations in concept expansion mode
$simMAX(c, r, q)$	Maximum concept-to-concept similarity score of concept c for resources r and q
$Scores(c, r, q)$	Candidate concept-to-concept similarity values of concept c for resources r and q
$descScore(c, r, q)$	Descriptor score of c to determine the recommendation score between r and q in concept expansion mode
$score_{exp}(r, q)$	Recommendation score of two resources r and q in concept expansion mode
$\Omega_{prefilter}$	Solution mappings of items resulting from prefilter graph pattern matching
Ω_{pre}	Solution mappings of relevant resources and annotations in prefiltered retrieval mode
$sim_{cond}(r, q)$	Conditional SKOS similarity of two resources r and q
$freq_{cond}(c)$	Frequency of c among all filtered resources
$IC(C_{cond})$	Conditional SKOS Information Content
P_{pref}	Graph pattern for preference-based querying
$R(Pr)$	Set of SKOSRec recommendations for profile Pr

$kmax\ score(Pr, q)_{q \in \Omega}$	Function that returns the k^{th} value of the list of recommendation scores sorted in descending order
$\Omega_{postfilter}$	Solution mappings for a postfilter graph pattern
Ω_{post}	Solution mappings in postfiltered retrieval mode
y	Aggregation-based recommendation
$R_{agg}(a)$	Set of aggregation-based recommendations for a preferred item a
$score_{agg}(y)$	Aggregation-based recommendation score of y
x	Sublevel recommendations
tp	True positive predictions
fp	False positive predictions
fn	False negative predictions
$prec$	Precision
rec	Recall
$f1$	F-Measure
mrs	Mean relevance score
$ndcg$	Normalized discounted cumulative gain
nv	User-based novelty
$prob(i)$	Likelihood to choose an item
nov_{pop}	Popularity-based novelty
$cos(i_l, i_m)$	Cosine similarity of two items i_l and i_m
$dist(i_l, i_m)$	Distance of two items i_l and i_m
$divC$	Content-based recommendation list diversity
$divU$	User-based recommendation list diversity
$JI(A, B)$	Jaccard index of two recommendation lists A and B

List of Equations

2.1	User-based similarity (CF)	13
2.2	Predicted preference score in user-based CF	13
2.3	Item-based similarity (CF)	13
2.4	Predicted preference score in item-based CF	13
3.1	Triple pattern	31
3.2	Evaluation of a triple pattern over a dataset	31
3.3	Conjunctive evaluation of two graph patterns over a dataset	31
3.4	Join of two solution mappings	32
3.5	Example RDFS rule	38
6.1	SKOS annotation triple	76
6.2	SKOS annotation dataset	76
6.3	SKOS annotations	76
6.4	Graph pattern to retrieve similar LOD resources	77
6.5	Multiset of mappings for similar resource retrieval	78
6.6	Shared concept set of two resources	79
6.7	IC of the set of shared concepts	80
6.9	Reduced item mappings	83
6.10	Reduced solution mappings of relevant resources and annotations	83
6.11	Informativeness values of common ancestors	86
6.12	Most informative common ancestors	86
6.13	Informativeness based on the number of descendants	87
6.14	Concept-to-concept similarity	87
6.15	Concept-to-concept similarity based on the hyP metric	88
6.16	Pondering function of the hyP metric	88
6.17	Set of proximate concepts	89
6.18	Union of all proximate concept sets	90
6.19	Concept expansion graph pattern	90
6.20	Concept expansion solution mappings	90
6.21	Descriptor score	91
6.22	Candidate score values	91
6.23	Maximum score value	91
6.24	Concept expansion recommendation score	92

6.25	Mapping of prefiltered resources	93
6.26	Prefiltered solution mappings of relevant resources and annotations	94
6.27	Conditional SKOS similarity	95
6.28	Conditional SKOS information content	95
6.29	User profile	97
6.30	Set of recommendations	99
6.31	Solution mappings for the postfilter constraint	99
6.32	<i>Postfiltered recommendation mappings</i>	99
6.33	Set of <i>aggregation-based recommendations</i>	100
6.34	Aggregation-based ranking score (MAX)	101
6.35	Aggregation-based ranking score (SUM)	101
6.36	Aggregation-based ranking score (AVG)	101
7.1	Precision	130
7.2	Recall	130
7.3	F-measure	130
7.4	Mean relevance score	130
7.5	Discounted cumulative gain	131
7.6	Normalized discounted cumulative gain	131
7.7	User-based novelty	131
7.8	Popularity-based novelty	132
7.9	Distance of two items	133
7.10	Content-based recommendation list diversity	133
7.11	Reduction ratio	136
7.12	Jaccard index	156

List of Listings

3.1	Example SPARQL query (simple matching)	30
3.2	Example SPARQL query (multiple matchings)	30
3.3	Expressive SPARQL query	37
3.4	SPARQL query for POI retrieval	38
3.5	SPARQL query after RDFS query rewriting	39
5.1	Example REQUEST query	58
6.1	Annotation property	75
6.2	Annotation query	76
6.3	Row query	78
6.4	Relevant resource query (<i>simQuery</i>)	79
6.5	Concept count	80
6.6	Estimation query	82
6.7	Query for optimized retrieval of relevant resources and annotations	84
6.8	Extraction of similar concepts	89
6.9	Query for <i>prefiltered constraint-based recommendation</i> retrieval	94
6.10	<i>Preference query</i>	98
6.11	Query for <i>postfiltered recommendation</i> mapping	100
6.12	Grammar of the SKOSRec query language	104
6.13	A simple SKOSRec query for the movie domain (Q1)	108
6.14	A SKOSRec query that triggers <i>flexible similarity detection</i> in the domain of DL search (Q2)	108
6.15	A simple SKOSRec query in the domain of DL search (Q3)	109
6.16	A SKOSRec query with a <i>prefilter</i> condition to retrieve Russian nature reserves (Q4)	110
6.17	The SKOSRec query of Listing 6.16 without the <i>prefilter</i> condition (Q5)	110
6.18	SKOSRec query with an <i>expressive prefilter</i> condition to retrieve Southeast Asian destinations (Q6)	111
6.19	SKOSRec query to generate <i>postfiltered recommendations</i> (aggregation-based) for a preference profile containing the city of London and related POIs (Q7)	112
6.20	SKOSRec query to generate simple <i>on-the-fly recommendations</i> for a preference profile containing the city of London (Q8)	112

6.21	SKOSRec query to generate <i>postfiltered recommendations</i> based on a <i>preference query</i> in the movie domain (Q9)	113
6.22	SKOSRec query to generate simple <i>postfiltered recommendations</i> in the movie domain (Q10)	114
6.23	SKOSRec query to generate <i>cross-domain recommendations</i> (Q11)	115
6.24	SPARQL query to generate <i>cross-domain recommendations</i> (Q12)	115
6.25	SKOSRec query to retrieve <i>cross-repository recommendations</i> for a single preference statement (Q13)	117
6.26	SKOSRec query to retrieve <i>cross-repository recommendations</i> from a preference profile (Q14)	118
7.1	SPARQL query to create user profiles	136

List of Definitions

1	Definition (RDF term)	28
2	Definition (RDF triple)	28
3	Definition (RDF dataset)	28
4	Definition (Triple pattern)	31
5	Definition (Evaluation of a triple pattern)	31
6	Definition (Annotation property)	75
7	Definition (SKOS annotation triple)	75
8	Definition (SKOS annotation dataset)	76
9	Definition (SKOS annotations)	76
10	Definition (Relevant resources and their annotations)	77
11	Definition (SKOS similarity)	79
12	Definition (SKOS Information Content)	79
13	Definition (Recommendation score)	80
14	Definition (Proximate concepts)	88
15	Definition (Relevant resources and annotations (concept expansion))	90
16	Definition (Descriptor score)	91
17	Definition (Concept-to-concept similarity score)	91
18	Definition (Recommendation score (concept expansion))	91
19	Definition (Prefiltered resources)	93
20	Definition (Annotations and relevant resources (prefiltered retrieval))	93
21	Definition (Conditional SKOS similarity)	94
22	Definition (Conditional SKOS Information Content)	95
23	Definition (Queried profile)	97
24	Definition (SKOSRec recommendations)	99
25	Definition (Postfiltered recommendation mapping)	99
26	Definition (Resources for aggregation-based retrieval)	100
27	Definition (Aggregation-based ranking score)	100

1 Introduction

This chapter will motivate the research agenda of this thesis. It will provide examples on how recommender systems (RS) are not yet fully equipped to handle certain personalization tasks and will elaborate on the hypothesis that a suitable application of Linked Open Data (LOD) can help to tackle this research gap (Sect. 1.1). Research objectives are derived from this hypothesis (Sect. 1.2). They comprise the specification and prototypical implementation of a LOD-enabled RS that improves and enhances existing approaches to recommendation retrieval. The chapter ends with a description of the thesis structure (Sect. 1.3).

1.1 Motivation

The amount of information has dramatically increased with the emergence of e-commerce applications and other online portals on the Internet [5]. Often, users browse many sites before they are able to decide on an item or service to consume. Whereas in the analog world gatekeepers (e.g., travel agents) assist in finding an offer that matches the customer's needs, in the online world, even for experts it is usually impossible to keep track of all available products without having access to software systems that can filter the data according to certain requirements. Users cannot process the wealth of information on the Internet on their own.

Fortunately, tools have been developed that help consumers find their way through the abundance of information. Besides search engines, another type of filtering technology can frequently be encountered. These are the so-called recommender systems. Researchers developed the first RS in the 1990s, but the technology is still highly relevant today. RS are tools that automatically generate suggestions based on known preferences of a person. Users express preferences through their interaction with the system (i.e., in their profile). The user profile contains information on how often a person has accessed an online article, or it stores the ratings a user has given for movie items he/she has already watched. Based on this data, RS generate personalized recommendations [204]. For this purpose, researchers have developed different approaches. Collaborative filtering (CF) algorithms are the oldest and most widely used method among them. They generate suggestions based on similar users. In these RS, users are regarded as like-minded, when they have expressed similar tastes for the same items [5]. The second largest group are content-based (CB) systems. CB algorithms calculate recommendations based on the similarity of items. These systems process metadata information to identify matching features [7].

All of these methods have in common that they rely on large amounts of data. A recommendation engine can only generate useful suggestions when it meets this requirement. Unfortunately, in real-world scenarios, it is often the case that the underlying database does not contain the necessary data. That is why users often receive either irrelevant or no suggestions at all (see Sect. 2.3). However, given the great importance of RS as tools that enable personalization, it would be desirable if, despite the missing data in the local database, engines could still generate recommendations.

It is one of the reasons why in recent years, researchers have started to explore ways to enhance available data sources with additional information from openly accessible and interlinked datasets (Sect. 5.2.2). The entirety of these public data collections is also known as the Linked Open Data cloud. It consists of a large number of repositories, which can be accessed over the web. The LOD cloud provides metadata descriptions for many real-world entities (e.g., multimedia items or travel destinations) that link to each other by utilizing the Resource Description Framework (RDF). RDF is a vocabulary language for structuring and publishing data sources in a graph-like fashion. The result is a network in which the nodes represent entities and the edges are the relations between these entities. Another advantage of this kind of markup is that it describes data semantically, i.e., links have a meaning. RDF graphs are machine-readable and can be freely used by software applications. Hence the data web is similar to the World Wide Web (WWW) because of its open accessibility and decentralized nature. Additionally, it can be processed without any manual interventions (Sect. 3.1). Over the last years, the LOD cloud has grown into a huge knowledge graph (Sect. 3.2), containing valuable information about items from many domains and data collections. It is a manifestation of the vision of a Semantic Web that was introduced by Tim Berners-Lee in 2001 [34]. The most prominent data collection in the LOD cloud is DBpedia. It contains structured information of Wikipedia articles. Hence, any third party can access data from the prominent online reference repository and make use of it [59]. Another positive feature of the data cloud is that it contains connections between data collections. For instance, some real-world objects occur in more than one LOD repository. Correspondence links between the collections identify matching objects. In this way, data can be queried and aggregated across repositories, such that software agents can discover new connections and interesting information (Sect. 3.2).

Given the abundance of metadata in the LOD cloud, it is not surprising that RS researchers have begun to harness these data sources to tackle data sparsity issues. Recommender systems operating at least partly on RDF resources are called Linked Data Recommender Systems (LDRS) or LOD-enabled recommender systems. Most existing LDRS are research prototypes. Some studies have shown that they can compete with traditional systems and help to overcome shortages in the local database. Researchers have applied content-based approaches by utilizing LOD metadata descriptions for offline computation of item similarities. This means that, prior to the actual process of generating recommendations, data must be extracted from the LOD cloud. Thus, the RS operator has to set up data models representing both the user pro-

file and the features of the items before the system runs. It may happen that user preferences are too simplistically represented in the system. Hence, despite the comprehensiveness and expressiveness of RDF graphs, metadata descriptions are still structured in the table-like format of attribute-value pairs. Although this approach is effective in terms of data enrichment (after all, it is possible to fill in the gaps in the local database), the consequence is that the central strengths of the LOD cloud are not fully exploited. The flat data structure reduces the richness of information from the RDF graph and prevents the full potential of LOD enhancement from being realized for recommender systems (Sect. 5.2.2).

Consider the following example for illustration. Suppose a user has stated that he/she likes a particular Indie rock band. A conventional LDRS would extract directly linked item features for the given band from the LOD cloud and determine similar bands from the same item feature table. However, it may well be that the descriptions contain irrelevant information, e.g., just the founding place or founding year of the band, instead of the style of music they usually play. Such metadata would not yield useful results in a purely content-based system and only achieve a few random hits. On the other hand, a recommendation request, which takes into account the band's position in the RDF graph, has a higher chance of identifying relevant suggestions. For instance, a graph-based query could explore the music items (e.g., albums or songs) released by the particular band and extract genre information for these items. Labeled links can be exploited to determine the respective descriptions in the RDF graph. The data web can answer such requests because the LOD technology stack provides the SPARQL Protocol and RDF Query Language (SPARQL) and suitable query engines that enable fast retrieval (Sect. 3.1.4). However, graph-based queries alone are not yet sufficient to generate personalized suggestions since they do not consider user preferences. What is missing is a tool that can combine graph-based query elements with similarity calculation. For this purpose, however, it is important that recommendations can be generated on-the-fly. Only then will a system be able to switch between processing of graph- and search-based query parts. To the best of the author's knowledge, there does not yet exist a recommendation engine that can perform these tasks on RDF data. Instead, current LDRS rely on a fixed set of item features, for whom values have to be extracted from the LOD cloud before suggestions can be generated. In this way, the strengths of graph-based querying, unfortunately, remain largely unused.

A purely SPARQL-based request, on the other hand, only returns results for graph patterns that exist in the repository. Consider the following example: Consumers may like to use recommendation engines that work on multiple domains simultaneously. For instance, a user, who has stated that he/she likes certain items from one domain may like to receive suggestions from another domain as well. In case a user profile contains feedback information for books, it would be desirable if the system could generate movie suggestions based on this data. While the approach of querying RDF graphs can be useful for this recommendation scenario, since it can identify matching objects based on entity type declarations or typed link information, it may also be the case that there are no direct links from a preferred book to a suitable movie in the

RDF graph. A possible solution would be to identify books that are similar to the ones in the user profile. In a subsequent processing step and through a graph-based query, the engine could explore whether the similar books have any connections to movie items in the repository. This approach can reveal implicit connections in the data and might help to return fewer empty result sets to the user. The idea of combined retrieval is not only relevant as a recommendation strategy, but also as a general (semantic) search method for LOD repositories because it unites the paradigms of query- and similarity-based retrieval. Therefore, the system should process the data online. This is because one processing step (e.g., identification of suitable movies through graph pattern matching) relies on the outcome of a preceding one (e.g., computation of similar book items). Almost none of the existing LDRS and LOD-enabled search systems address these issues, which prevents effective usage of available knowledge sources for retrieval tasks (5.2). The LOD cloud stands out for its open availability, the wealth of data sources and the expressive power of the RDF data model. On the other hand, however, accessing LOD repositories also has some challenges. It is a problem that many collections utilize different RDF-based vocabularies. This causes data heterogeneity and complicates *on-the-fly retrieval* since the engine might not be able to process all types of LOD resources right away (Sect. 3.3).

Only a few of the vocabularies in the LOD cloud have become a de-facto standard by now. The usage of such standards is a vital prerequisite for retrieval from different collections. Fortunately, there exist vocabularies that are widely used as well as applicable to ad-hoc item-to-item similarity calculation. In this context, a key standard is the Simple Knowledge Organization System (SKOS) vocabulary. SKOS vocabulary concepts often annotate items in LOD repositories and, thus, help to describe real-world entities. For instance, in the DBpedia repository, an LOD resource representing a book is characterized by subject descriptors (e.g., stating the topic or author of the book). The descriptors are part of the SKOS-based DBpedia category graph. SKOS subjects are uniquely identified LOD resources. Therefore, they can be conveniently processed. Another advantage is that many SKOS vocabularies provide expressions for knowledge representation. Hence, they declare the meaning of concepts and organize them in a semantic network. These links comprise hierarchical and (non-)hierarchical connections and can be exploited for similarity calculation as well (Sect. 6.1).

Another challenge of LOD-enabled *on-the-fly recommendations* is that the engine needs to process many records simultaneously. In 2016, the LODStats crawl identified 290 billion triple statements describing more than 54 million entities in the LOD cloud [149]. While a recommendation request will only query a tiny subset of the data web, similarities might still have to be computed for many item pairs. Thus, methods to speed up retrieval need to be investigated. A further challenge of LOD processing is that resources reside in different repositories. Additionally, they may be spread over several physical servers. For instance, the 2016 LODStats crawl found that LOD resources can be found in more than 2900 datasets. However, in the context of personalized search, users are often interested in obtaining results for a specific topic rather than from a particular collection. Since LOD resources often describe the same real-

world objects in different datasets, *cross-repository recommendations* are a highly desirable feature. To the best of the author's knowledge, there does not yet exist an LOD-enabled RS that thoroughly addresses these challenges of LOD processing (Sect. 5.2).

1.2 Research Objectives

The previous paragraphs have shown that there is currently a research gap regarding effective application of LOD for recommendation tasks. Aside from the use of LOD repositories for metadata enrichment, existing RS neither address the strengths of graph-based RDF data nor the challenges of LOD processing. It is a significant shortcoming since the LOD cloud holds rich information sources on all kinds of topics, which should be accessible through end-user applications. Therefore, novel approaches and technologies are needed. It was precisely the vision of Tim Berners-Lee, the presumed inventor of the WWW, that software agents can work seamlessly on a web of interconnected data [34]. His notion of the Semantic Web has partly been realized with the emergence of the LOD cloud during the past years. However, many of the tools that can make use of this data have yet to be developed.

The application of RDF data for current RS is limited to simple data enhancement, where LOD is fed into an engine's local database to account for issues of data sparsity. What is missing so far is an RS that can process expressive, graph-based recommendation requests over large triple stores. In addition to the strengths of LOD usage for recommendation tasks, the challenges of RDF data processing have yet to be solved as well. Again, there is a lack of tools and techniques that deal with phenomena such as data quantity, data heterogeneity or data distribution. The research on LOD-enabled RS is still in its infancy and stuck in concepts of table-structured data management commonly applied in relational databases. The adaptation of existing content-based algorithms to the characteristics of the LOD cloud has not yet entirely taken place. While there are a few systems that already address aspects of the above-described features of publicly available RDF data, currently there is no LOD-enabled RS that realizes the full potential of LOD resources for recommendation tasks (Sect. 5.2).

Hence, the research goal of this thesis is the **development of a LOD-enabled recommender system that takes into account the strength and challenges of Linked Open Data to improve existing recommendation approaches**. The use of RDF data is not an end in itself. The to-be-developed system should deliver recommendations that are at least comparable in quality, if not better, than suggestions produced by existing LOD-enabled content-based methods.

From the overall research objective, three subordinate goals are derived.

1. Requirements specification for a LOD-enabled recommender system which addresses the strength, challenges and technical requirements of LOD processing.
2. Prototypical implementation of a LOD-enabled recommender system meeting the demands of the requirements specification.

3. Evaluation of the system's performance (processing times, workloads, and recommendation quality) in example usage scenarios.

The implementation and subsequent evaluation ensure that the prototype satisfies the previously defined requirements. The analysis will identify example use cases to demonstrate the general applicability of the engine (see Sect. 6.2).

1.3 Thesis Outline

The thesis is structured as follows: Chapter 2 gives an overview on recommender systems. It characterizes their purpose, their basic functionalities and the type of data sources that are usually processed by these systems. Additionally, the chapter describes the most widely used recommendation algorithms, namely collaborative filtering, content-based, hybrid and knowledge-based approaches and presents the limitations of these algorithms.

Following this, Chapter 3 discusses how Linked Open Data could help to address some of the weaknesses of existing RS. Thus, after a brief overview of the origins and basic components of the LOD technology stack (e.g., RDF and SPARQL), the potential and challenges of LOD processing in the context of recommendation retrieval are described in detail. Building on the central findings of Chapters 2 and 3, Chapter 4 compiles a requirements specification for an RS that reflects the previously specified issues in the form of clearly defined system functionalities and properties. In addition to considering the central strengths and weaknesses of RDF processing it takes into account the technical features of the LOD cloud.

In the related work section (Chapter 5) it is investigated whether and how existing real-world recommender systems or LDRS research prototypes already provide any of the major features of the requirements specification. In this context, solutions of adjacent retrieval paradigms, such as query-based RS on relational databases or faceted search are taken into account as well.

In Chapter 6, suitable usage scenarios are described, based on which the system's novel recommendation techniques are explained. The approaches implement the requirements specification. They are based on the state of the art of RS, LDRS and retrieval systems in general. The developed system utilizes SKOS vocabularies and is therefore called SKOSRecommender (SKOS-Rec). In addition to a SKOS-based calculation of item similarities, the system implements other features, such as *(fast) on-the-fly retrieval*, a SPARQL-like query language, and options to insert processing steps of similarity calculation at different stages of the recommendation workflow. Chapter 6 describes each one of these system features in detail.

Chapter 7 focuses on the evaluation of the SKOSRec engine in the previously defined usage scenarios. In this context, it is determined whether the system provides quick responses to recommendation requests, and also whether it generates relevant suggestions. For this purpose, the SKOSRec engine was evaluated with regard to different quality dimensions (i.e., *accuracy*, *novelty*, *diversity* and *perceived usefulness*) in offline and online experiments using actual user

preferences. Chapter 7 explains the methods and results of these studies.

Finally, Chapter 8 provides the key findings of the thesis. It will be clarified whether the novel system features of the SKOSRec engine can provide added value for users as well as meet the demands of the requirements specification. The chapter will also discuss limitations of the findings and future research directions.

2 Recommender Systems

After having presented the central research question and goals of the thesis, Chapter 2 outlines the general characteristics of recommender systems (Sect. 2.1). It will describe the key approaches to recommendation retrieval, namely collaborative filtering, content-based, knowledge-based and hybrid recommender systems (Sect. 2.2) and will summarize the limitations of the different techniques (Sect. 2.3).

2.1 General Characteristics of Recommender Systems

Recommender systems are an integral part of today's Internet landscape. Major e-commerce sites, such as Amazon.com [11] or Netflix [172] cannot be imagined without an RS component. For instance, users who have watched and liked a series or a movie on Netflix will receive suggestions for other related films in a separate section on the screen, in which all items that are similar to the previously preferred ones are listed. As a second example, think of an Amazon customer who has bought a particular book. Based on this past purchase, the e-commerce retailer will recommend additional books under a subheading that refers to „Customers who bought this item also bought...“. End users only see the results of an RS's key task: finding the items which are most suited to be recommended to a customer [121]. The process of automatically generating suggestions is at the center of RS research.

The need to automate recommendation retrieval arose at the beginning of the 1990s when the first e-commerce applications and other Internet sites offered their services online. In these formative years, providers were trying to come up with solutions to handle increasing quantities of digital resources and to ensure that users found suitable items [5]. The explosive growth of digital content can still be witnessed today and is known under the term „information overload“. In this context, recommender systems have proven to be effective tools that help users to navigate tremendous amounts of online information [204].

Back then, the starting point of RS research was to mimic social interaction. Hence, system developers focused on the observation that users tend to follow recommendations proposed by their peers or a trusted individual when faced with a decision. A user, who does not know which books or movies to consume next, might find it useful to consider suggestions by a friend or to consult a critic's list of recommendations in a magazine to make an informed decision [204]. A software system often has to identify the right items from hundreds of thousands of products. In this context, a straightforward approach would be to focus on best-selling items or most viewed

items. However, this strategy does not at all address individual customer needs [121]. Additionally, it always refers customers to the same set of products, even when they have completely different preferences. Although popularity-based approaches can be helpful under some circumstances, they are often not perceived as useful enough to trigger a purchase. Therefore, RS researchers focused on personalization early on, whereby the engine bases recommendations on a user's previous interaction with the system. Behind this approach lies the assumption that past preferences (e.g., as shown by purchase or by click behavior) are reliable indicators for future choices. The prevalence of personalized RS on the Internet is evidence for this hypothesis [7]. Since the early systems, RS have fundamentally advanced the browsing experience in online applications. On many platforms and besides general search functionalities, they are one of the key entry points to a site's content. Hence, the development of recommender systems is often driven by application-specific demands [137].

RS have gained increasing interest during the past years as an independent research field both in industry and academia. They are components of important highly frequented Internet sites, and there exists a multitude of conferences, workshops, and journals that publish research articles on the topic. The intertwining of industry demands and academic research in this field also became evident when the streaming site Netflix held a competition for improving their recommender system. The company invited researchers to develop an algorithm that predicts user preferences for their movie items. They granted a 1 million dollar prize to the winning team. Within the scope of this competition, the company published a historical dataset that was used by researchers to tune algorithms [204].

These examples illustrate the profound practical impact of research on recommender systems, where improvements directly benefit both customers and service providers. On the side of the service provider, the overarching goal is to increase product sales. Therefore, it is most natural that retailers intend to boost the user experience. In addition to personalizing search tasks, the purpose of RS is the promotion of products which might be otherwise hard to find. From the application provider's point of view, it is also vital that the engine supports customer loyalty. In this regard, a well-functioning recommender system can help to ensure that customers frequently return to the site to seek assistance when choosing products [107, 204].

Customers, on the other hand, primarily want to *find good items* with the help of the RS [107]. This kind of retrieval task is also known as the *top-k recommendation problem*, where only a small fraction of items is displayed to the user. Therefore, the absolute values of predicted user preferences are irrelevant, because only the relative item order constitutes the composition of the recommendation list [7].

In some cases, however, users may want to *find all good items* that fit a certain query, e.g. when the suggestions concern a crucial application context, such as the recommendation of medical treatments or financial products. Additionally, RS may sometimes be useful to provide *annotations in context*, such that users are assisted when deciding which items of a product series are relevant (e.g., when selecting shows of a TV program). In other cases, users may want to obtain

recommendations for a multitude of items. In this kind of mass retrieval case, typical scenarios are *recommendations for a sequence* of items (e.g., music tracks) or *recommendations for a bundle* of products that can be grouped together (e.g., suggesting a book and its movie adaptation). Additionally, RS can facilitate *product browsing* and explorative search tasks. Even when users are not looking for a particular item, RS assist with serendipitous retrieval forms thus helping users to discover unexpected items [107, 204]. Besides the multitude of potential tasks, this thesis will focus on the *find good items task*, as it is the most common scenario [107]. Certain object types are involved in a typical recommendation workflow. Since RS provide personalized suggestions to users, they have to keep track of users, items, and user-item-interactions within the system [204]. One of the most natural forms of user modeling is to store past preferences as a list of ratings or as a session history log [121]. Additionally, a profile can contain sociodemographic attributes (e.g., age, gender or profession). Items are objects that a user might consume, visit, lend or purchase (e.g., a news article, a travel destination or a music album). As is the case with users, items have to be represented by specific features (e.g., the genre of a movie or the author of a book). These features distinguish them from other items in the system [204].

It is vital to the retrieval process that the engine has access to past user transactions. RS researchers usually differentiate between two types of preference declarations. Users can express their likings either explicitly or implicitly. Explicit feedback often has the form of a numerical rating on a 1-5 scale, but users can state it as an ordinal, binary (e.g., „good“ or „bad“) or unary rating (e.g., a Facebook „like“) as well [7, 121, 204]. However, consumers are usually reluctant to provide explicit feedback because of the extra work required. That is why system providers often use implicit feedback from a user's past interaction with the system. It is assumed that by processing data from purchasing transactions or by evaluating click behavior, user tastes also become apparent [7, 40].

Since the development of the first systems, RS technology has spread to many application domains. *Entertainment RS* are among the most widely encountered systems. They suggest movies, books, music acts or TV programs to users [40]. E-commerce platforms also recommend entertainment articles. However, since these applications suggest items from other product categories as well (e.g., electronics articles), RS components on commercial sites are grouped into the category of *E-Commerce RS*. Another type of engines are *Content RS*. These systems suggest digital documents, such as news/blog articles, web pages or scientific publications. When a system recommends items that refer to services (e.g., travel activities), it falls into the category of *Service RS*. Recommendations can also be made in a social context. For instance, suggestions in online social network communities help users to find other people they might like to befriend. These systems are subsumed under the term *Social RS* [204].

2.2 Recommendation Algorithms

2.2.1 Collaborative Filtering Recommender Systems

Given the diversity of application domains and the abundance of RS both in industry and academia, it is astonishing that the underlying algorithms can be boiled down to a few distinctive approaches. One of the key methods is collaborative filtering. CF engines were the pioneering recommender systems. The first of this kind was the GroupLens system. It automatically suggested Usenet News articles [137, 201]. Another one of this early RS was the „Ringo system“ that recommended music artists [137, 225]. The researchers applied CF techniques but did not call them by this name. Instead, they used the phrase „word of mouth“. This term describes the technique fairly well: In CF systems, the real-world phenomenon is mirrored by identifying like-minded peers through their ratings within a user community. Tastes of similar peers are applied to predict whether a certain user might like an item he/she has not yet rated [5]. Aside from the assumption that past preferences determine future choices, this technique thrives on time-independent user tastes. In its most basic form, CF works as follows: User ratings for items are stored in a matrix. The matrix serves as input for the CF-based RS. The system predicts a numerical value that indicates a user’s potential interest for specific items. Based on these results, a *top-k recommendation* list is generated that contains the items with the highest preference scores [121].

Table 2.1: Example user-item rating matrix

	Inception	They Call Me Trinity	The Terminator	Bridget Jones’s Diary
Alice	5		4	2
Bob		3	5	
Carol		5	5	
Dave		4	4	2
Eve	3	2	1	5

Table 2.1 depicts an example user-item rating matrix. It contains known preferences of users for certain movies in the database. For example, the 5-star rating of the user Alice for the movie „Inception“ indicates that she strongly likes this item. Based on the matrix, the engine predicts preferences for yet unrated movies. For instance, the engine would try to infer Alice’s rating for the western movie „They Call Me Trinity“. If the calculation reveals a strong preference for a particular movie; the item would show up in the final recommendation list. The preference score is based on the most similar users (often referred to as peers or nearest neighbors) as well as on the past rating behavior of these users.

The example can be formally described as follows: The set of users is denoted as $U = \{u_1, \dots, u_n\}$. Movie items are represented as the set $R = \{r_1, \dots, r_n\}$ and user preferences are stored in a rating matrix $Pref$. For a particular user a , the engine identifies other users with similar rating

behavior. For this purpose, common similarity measures, such as the *pearson correlation coefficient* (see Eq. 2.1) or the *cosine similarity* metric are applied. Like-mindedness between a user a (e.g., Alice) and a user b (e.g., Bob) is determined from ratings. However, one user might, on average, give lower ratings than another one, even though their tastes are more or less congruent. To account for this potential bias, Equation 2.1 contains the average rating for each user (e.g., $\overline{pref_a}$) as well [121].

$$userSim(a, b) = \frac{\sum_{r \in R} (pref_{a,r} - \overline{pref_a})(pref_{b,r} - \overline{pref_b})}{\sqrt{\sum_{r \in R} (pref_{a,r} - \overline{pref_a})^2} \sqrt{\sum_{r \in R} (pref_{b,r} - \overline{pref_b})^2}} \quad (2.1)$$

After having obtained user-based similarity scores, the engine selects a user's most similar peers and makes predictions for specific user-item pairs (e.g., Alice's rating for the movie „They Call Me Trinity“). For this purpose, the system includes preference information from the user neighborhood (Nb). The more similar in taste a neighbor is, the more his ratings are taken into account for the final prediction (Eq. 2.2).

$$pred(a, r) = \overline{pref_a} + \frac{\sum_{b \in Nb} userSim(a, b) \cdot (pref_{b,r} - \overline{pref_b})}{\sum_{b \in Nb} userSim(a, b)} \quad (2.2)$$

Even though CF algorithms achieve good results, their application has some challenges. Major e-commerce sites handle millions of users and therefore need to scan large amounts of data for neighborhood selection. Additionally, CF systems require that items have been rated by a sufficient number of users to predict future choices reliably. However, the profile overlap among users is usually low, due to the sparsity of the user-item matrix. Therefore, some large online retailers often revert to the method of *item-based recommendation* [121]. It is based on similar items instead of on similar users. The idea behind this technique is to recommend objects that are similar to items a user has liked in the past [5, 212]. The item-based approach compares the rating vectors of each item. For instance, to predict a user's (u) preference for the movie „They Call Me Trinity“, the system searches for other items in the database with similar ratings (e.g., „Inception“ or „Terminator“) that were given by all users (U) (Eq. 2.3).

$$itemSim(i, j) = \frac{\sum_{u \in U} (pref_{u,i} - \overline{pref_u})(pref_{u,j} - \overline{pref_u})}{\sqrt{\sum_{u \in U} (pref_{u,i} - \overline{pref_u})^2} \sqrt{\sum_{u \in U} (pref_{u,j} - \overline{pref_u})^2}} \quad (2.3)$$

The final preference score of a user u for an item r is based on the similarity of this item with the ratings of u for other objects (Eq. 2.4).

$$pred(u, r) = \frac{\sum_{i \in ratedItems(u)} itemSim(i, r) \cdot pref_{u,i}}{\sum_{i \in ratedItems(u)} itemSim(i, r)} \quad (2.4)$$

In addition to the classification of CF approaches into user-based and item-based algorithms, further distinctions can be made. For instance, a common categorization divides CF techniques either into memory-based or model-based approaches. The beforementioned methods of nearest neighbor selection fall into the category of memory-based algorithms. These approaches process preference information directly, such that new ratings update result lists. In contrast, memory-based approaches precompute a model from rating data. Models can quickly become outdated when new ratings enter the database. However, these approaches provide quick responses, because models are already present before recommendation retrieval. Common techniques in this category of RS are matrix factorization, association rule mining or Bayesian classification [40, 121].

2.2.2 Content-based Recommender Systems

While CF algorithms rely on rating data, content-based approaches combine user preference information with item metadata. This approach is especially helpful when score values are incomplete. Table 2.1 shows that the movie „Inception“ does not have many ratings. Therefore, the engine would most likely not recommend it, although some users might be interested in it. In this regard, item descriptions (see Tab. 2.2) represent a possible solution [7]. A content-based approach would reveal that users, who liked the movie „The Terminator“ in the past might also be interested in the movie „Inception“, since both movies share some characteristics, such as the genre (i.e., „Science fiction action films“). Whereas with a common CF approach, a user like Bob would not have received a suggestion for the movie „Inception“ due to missing data, content-based RS can counterbalance issues of rating sparsity [5]. A CB system does neither need explicit feedback nor a huge user community to generate suggestions [152]. As long as the system has information regarding past preferences, a user profile can be generated. In combination with item descriptions, user profiles facilitate content-based retrieval [121].

Metadata characterizes an item within a certain domain. For instance, in the movie domain a feature set typically consists of the director, the starring actors, and movie genres (see Tab. 2.2). Item descriptions are usually available in text form. In some domains, such as online retailing, the manufacturers of the goods often provide the text content in a table-like format. In these cases, the engine can easily process feature information. In other cases, however, item descriptions have to be extracted from the raw text, e.g., when processing news articles or scientific publications. Therefore, it is common to apply natural language processing (NLP) techniques to extract the most important keywords [7, 121].

Once features have been identified, the easiest way to find similarities between an item in a user profile and another one is to measure the overlap of their keywords [121]. Content-based similarity detection closely resembles information retrieval (IR) approaches. In typical IR settings, such as full-text retrieval, the engine matches search keywords against feature indices of document collections. However, preferences are expressed differently in CB and IR systems. CB

Table 2.2: Example item features of DBpedia movies [245]

	Director	Starring	Subjects	Music
Inception	Christopher Nolan	Leonardo DiCaprio; Joseph Gordon-Levitt	American films; Heist films; Science fiction action films	Hans Zimmer
They Call Me Trinity	Enzo Barboni	Terence Hill; Bud Spencer	Western genre comedy films; Spaghetti films	Franco Micalizzi
The Terminator	James Cameron	Arnold Schwarzenegger; Linda Hamilton	American films; Science fiction action films; Time travel films	Brad Fiedel
Bridget Jones's Diary	Sharon Maguire	René Zellweger; Hugh Grant	British films; Romantic comedy films	Patrick Doyle

systems infer recommendations from the user profile. Search results of IR engines, on the other hand, are generated by queries, which contain certain keywords. Despite this, both system types apply the same techniques to identify relevant items. In IR research it has long been known that text snippets need to be weighted to account for the fact that keywords do not describe a document equally well [158].

CB systems apply common keyword weighting methods from IR. One of these techniques is known as *term frequency-inverse document frequency* (TF-IDF). It projects term occurrences to a multidimensional Euclidian space, from which similarity values are determined. Let $K = \{key_1, key_2, \dots, key_n\}$ denote the set of available keywords. The weighted document vectors are defined as $\vec{d}_r = \langle w_{1,r}, w_{2,r}, \dots, w_{n,r} \rangle$, where $w_{m,r}$ denotes the weight of keyword key_m in document d_r representing item r [152].

The idea of TF-IDF weighting is that the more frequent a term is, the better it is suited to describe a document's content. However, a sole consideration of term frequency would corrupt recommendation results in favor of items that are described by more keywords. Therefore, frequency values are usually normalized with regard to the maximum frequency $max_z freq(z, r)$ within a document (see Eq. 2.5) [121, 152].

$$TF(key_m, d_r) = \frac{freq(m, r)}{max_z freq(z, r)} \quad (2.5)$$

Term occurrence is not the only indicator of keyword importance since many terms might not discriminate well between documents. For instance, in Table 2.2, the term „film“ frequently appears in the metadata description of the movie „Inception“. It is a rather general term in the

movie domain. Therefore, despite counting term occurrences, the *inverse document frequency* (IDF) of a keyword is additionally important. IDF measures the ratio of term appearances ($freq(m)$) among all recommendable items N in a database (Eq. 2.6). The TF-IDF values are determined by multiplication of *term frequency* values with *inverse document frequency* scores [152]. The weighting scheme assigns a high TF-IDF value, whenever a term frequently occurs in an item's description, but rarely appears in other item descriptions of the document collection. Hence, a high TF-IDF score indicates that a keyword is well suited to describe an item.

$$IDF(key_m) = \log \frac{N}{freq(m)} \quad (2.6)$$

$$TFIDF(key_m, d_r) = TF(key_m, d_r) \cdot IDF(key_m) \quad (2.7)$$

After having obtained TF-IDF values for item feature sets, similarities can be calculated. Typically feature vectors are used in conjunction with the *cosine similarity* metric. As for user-based algorithms, this measure facilitates neighborhood selection. In contrast to CF systems, however, the similarity is calculated using TF-IDF values instead of being based on rating data. For each item pair, their weighted feature vectors d_i and d_j are taken into account (Eq. 2.8) [152].

$$docSim(d_i, d_j) = \frac{\sum_{m \in K} w_{m,i} w_{m,j}}{\sqrt{\sum_{m \in K} w_{m,i}^2} \sqrt{\sum_{m \in K} w_{m,j}^2}} \quad (2.8)$$

Afterward, the engine selects the most similar items (i.e., nearest neighbors) for preference prediction. For this purpose, it applies a slightly modified version of Equation 2.4. Then, it weights user ratings with similarity scores of each item from the nearest neighbor list [121].

2.2.3 Knowledge-based Recommender Systems

The previously introduced approaches of collaborative and content-based filtering are the two most distinct types of RS mentioned in the literature [5,40,137,152]. Nevertheless, there are further approaches to recommendation retrieval. One of them is the category of knowledge-based (KB) recommender systems. These systems rely on a data repository that contains domain knowledge. KB engines are independent of rating data. Therefore, they can still provide recommendations even when new items or users enter the system (i.e., *cold-start problem*). Common content-based and collaborative filtering RS, on the other hand, are severely affected by data sparsity issues [7].

Additionally, item- or user-based recommendations might not suffice in certain application domains. For instance, in areas with highly complex or customizable products (e.g., real estate, automobiles, financial services or computers) users prefer to state their requirements directly and therefore need to query specific sets of item properties [7, 121]. Products in higher price

segments are rarely bought and often subject to frequent changes. For instance, in technical domains, such as personal computer (PC) sales, preferences regarding processor power or main memory storage quickly become obsolete over a period of several years. Therefore, past user ratings are not as relevant for these domains as they are in other application areas [7].

While for CB and CF systems knowledge acquisition stops when the engine has obtained user ratings/item features, this process is more laborious for knowledge-based RS. Aside from maintaining detailed item descriptions, these systems often define additional rule sets. Rules either specify how user requirements ought to be matched with product features or declare compatibility constraints for allowed combinations of item properties [121]. For instance, when users require their computers to have advanced technical features (e.g., dozens of gigabyte [GB] of random-access memory [RAM] storage), it is sometimes not feasible to recommend machines from lower price segments. Therefore, KB systems heavily depend on the provision of domain knowledge. Another characteristic of knowledge-based RS is that the preference elicitation process is a key component of the recommendation workflow. Since users specify their needs in great detail, systems need to provide interactive feedback mechanisms with which they can subsequently get to know the available items. The interfaces that are required for these interactions greatly depend on the algorithm used by the system [7].

Knowledge-based RS are commonly grouped into two major categories. One of them is constraint-based RS. These systems match user requirements with the item features in the database. An example is the specification of a maximum price for a PC. The engine queries all products that fulfill this constraint. A constraint-based request consists of a couple of such specifications in the form of a conjunctive query. It comprises a set of conditions that are connected by AND operators. The solution set to this request contains all items of the product table that fit the selection criteria [121]. Constraint-based RS regard recommendation tasks as constraint satisfaction problems. They determine solutions according to a set of variables, finite domains and acceptable value combinations. Additionally, compatibility constraints define permissible instantiations of customer requirements (e.g., if 12 GB main memory is required for a desktop PC, the price should not be below 600 Euro). Filter conditions specify the circumstances under which the engine should select certain products (e.g., only PCs with high-end graphics boards should be presented, upon receiving a request for a gaming PC). Aside from an internal logic on how to select items according to user specifications, a constraint-based RS should provide an interface, with which users can modify, add or relax their requirements during a retrieval session. The interface should help users to learn enough about the available items. For instance, it can present default options or property lists, with which consumers start a series of retrieval requests [121].

The second major type of knowledge-based RS is case-based recommenders. They retrieve items according to a given similarity metric. These metrics quantify the extent to which the items in the database match the user specifications. Users start the retrieval process by specifying their favorite target cases. Similarly to CB RS, the engine identifies recommendation

candidates through item attributes. However, the applied metrics are more advanced, since the knowledge base often contains heterogeneous data sources (e.g., categorical vs. numerical attributes, symmetric vs. asymmetric utility functions, properties to maximize vs. properties to minimize). Hence, the configuration of such a measure is laborious and largely domain-dependent. Additionally, the recommendation process does not stop with similarity calculation. Once similar items have been found, users refine their requests by tuning certain properties. This process of subsequent refinement is called critiquing and has to be supported by a powerful user interface as well. Unlike constraint-based RS, case-based RS do not as easily produce empty result sets, because requirements are not formulated as hard constraints. Rather, they help users in gaining an understanding of the item space through interactive browsing [7, 121].

2.2.4 Hybrid Recommendation Approaches

The previously presented approaches to recommendation retrieval rely on certain data sources to provide automated suggestions. CF systems are based on user ratings, while CB systems utilize metadata descriptions. In addition to user profiles and item features, KB engines require a repository of domain knowledge [121]. Thus, each of the presented approaches has problems when dealing with limited data sources. While CF systems perform poorly when rating data is sparse, content-based recommendations tend to be worse as soon as item descriptions are of low-quality [40]. Knowledge-based systems, on the other hand, require a repository of domain knowledge which often needs to be manually maintained and is therefore prone to errors [7]. Besides this, they contain item metadata in relational form and thus need to process heterogeneous data sources [121].

Hence, neither of the presented approaches is superior to another one. Rather, it depends on the available data sources and the particular application domain whether a certain method performs well. Sometimes it might be even better to combine approaches to overcome the limitations of the methods. These algorithms are called hybrid recommender systems. They fuse different paradigms to boost recommendation quality [5, 40]. One frequently encounters mixings of CF and CB techniques among the hybrid approaches [5]. However, it is also possible to take advantage of similar recommendation approaches (e.g., various CF algorithms) and combine them within a single model [214].

Hybrid recommendation approaches are as manifold as recommendation algorithms. Several authors have presented categorization schemes for hybrid RS [7, 45, 121]. For instance, Jannach proposes grouping approaches into monolithic, parallel and meta-level hybridization designs. A monolithic hybrid RS combines feature data from different recommendation paradigms. It is done by either mixing item features directly (e.g., by using ratings and text-based features in the nearest neighbor algorithm) or by applying a feature augmentation hybridization technique. In the latter case, the engine feeds results from one recommendation paradigm into an algorithm of another paradigm (e.g., as in content-boosted CF systems) [121].

Parallel hybridization techniques, on the other hand, run several recommenders concurrently and aggregate results after completion of the separate retrieval processes. The aggregation of the results can be carried out by assigning weights to different scores. It is also possible to generate separate recommendation lists whose results are then selected according to previously defined rules. For instance, the problem of new and unrated items can be addressed by resorting to CB recommendations until the amount of rating data is sufficient enough to provide helpful suggestions. This parallel hybridization technique is called switching. Another parallelization technique is the mixed representation, where the RS presents different recommendation lists side by side within a single user interface. By this means, the user can compare the rankings and decide which of the suggestions suits his/her interests most [121].

Another major category of hybrid RS is algorithms that apply meta-learning strategies (i.e., ensemble methods) [69]. Meta-level methods utilize out-of-the-box approaches to refine recommendation scores in a cascading fashion [7]. The system usually combines the results of straightforward algorithms into an aggregated prediction model. A famous example of an ensemble learning technique is „BellKors Pragmatic Chaos“, which was the winning solution of the Netflix Prize in 2009. It applied a boosting strategy that trained several base learners to minimize error rates [139].

The presented hybridization designs attempt to overcome limitations of existing recommendation paradigms. However, while hybrid RS often improve accuracy [139,249,256], they require additional computational resources and engineering efforts. Hence, major online retailers, such as Amazon or Netflix often hesitate to apply a hybrid recommendation strategy and rather rely on a scalable conventional approach [147,213].

2.3 Limitations of Recommender Systems

The previous sections have presented different recommendation paradigms. The approaches work most efficiently when being utilized in a suitable application scenario. Hence, online providers, such as commercial retailers, should choose a method according to the specific requirements of their application as well as based on the quantity and quality of available data sources. For instance, CF systems work best when the amount of rating data is sufficiently high. When users have provided enough preference information, CF systems can give suggestions across genres. They are also suited for items that do not have a text description [5,45]. Additionally, CF-based suggestions do not require domain knowledge, and their quality often evolves as new ratings tend to sharpen profiles [45].

On the downside, CF systems are heavily dependent on a critical mass of user ratings. Many users are reluctant to give explicit feedback. Often, catalogs contain a significant amount of items, of which many do not have a single score attached to them (*data sparsity problem*). Even if there are enough ratings, the CF engine cannot identify similarities, whenever a new item

(*new item problem*) or a new user (*new user problem*) enters the database. Thus, the system might never recommend certain items even though there are users who would have an interest in them [131]. CF systems are also susceptible to the *lemming effect*. The effect occurs when the engine recommends items in a self-perpetuating cycle. Hence, users with niche interests would most likely not find what they are looking for since they do not share interests with a critical mass of like-minded peers [45].

CB systems, on the other hand, draw their strength from item descriptions and can be therefore tailored to special interest domains. Like CF recommenders, CB systems do not require an elaborate knowledge base as long as user preferences can be related to the items in the database. Similarly, suggestions from CB engines tend to get better over time.

The shortcomings of CB RS can be attributed to aspects of data quality and quantity accordingly. New users, who have not yet set up a profile, will not be able to receive any suggestions [45]. Additional problems arise due to insufficient metadata descriptions. Whenever the quality or quantity of item feature information falls below a certain standard level, RS performance deteriorates (*limited content analysis*) [5]. For instance, it is vital that keywords that refer to the same entities are uniquely identified throughout a collection [181]. In a CB book RS, author names should have an authority record entry. Thus, it is avoided that different text expressions, which refer to the same entity (e.g., „J.K. Rowling“ vs. „Joanne K. Rowling“), distort the calculation of item-to-item similarities. Another example of *limited content analysis* is that of too little metadata content. With only a small set of item features, two rather unrelated items may be described by the same keywords [5].

Another shortcoming of CB systems is the *overspecialization problem*. Although reliance on user interests is useful, it bears the risk of suggesting items that are not new to the user. Whereas in CF systems unexpected items have a higher chance of being identified, due to the diversity of like-minded peer groups [5], CB systems can only resort to metadata information that might point to items the user already knows of [131].

KB RS, on the other hand, can tackle non-product related aspects, e.g., by integrating contextual factors, such as a vendors reliability (when buying a camera) or trending neighborhoods (when purchasing a property) into the utility functions of their knowledge base [45]. Therefore, KB systems are highly adaptive to customer needs and not as dependent on user preference information and item feature data. In contrast to CF and CB systems, common *cold-start problems* do not arise with KB engines as they obtain user preferences at runtime [7]. Usually, providers apply KB engines in high-end product domains with infrequent purchases. As preferences for higher priced products or services may change over time, historical datasets are not as valuable for KB systems as for CF and CB engines. However, the knowledge engineering process often requires manual creation of domain rules, which can be a laborious and error-prone undertaking. On the user side, the required effort is equally high. Users might have to complete several feedback rounds until they have set up their final set of preferences [45].

An additional problem of KB systems is their application-specific focus. It is most likely that

the knowledge base for a particular product type cannot be transferred to other domains, while CF or CB algorithms often can be utilized throughout different applications [7]. The combinations of the paradigms in a hybrid approach might compensate for some of their shortcomings as well as increase recommendation quality. However, hybrid RS also require considerable tuning effort and computational resources. Table 2.3 summarizes major strengths and weaknesses of the recommendation paradigms.

It is possible to address some of the previously outlined weaknesses of RS, such as the *new item problem*, the *overspecialization problem* or the problem of *limited content analysis* by enhancing the local data with additional information from the LOD cloud. The data web offers valuable sources from various domains that are openly available through public interfaces and could be applied to improve content-based RS. In addition to the provision of metadata, the LOD cloud might also facilitate more flexible and adaptive retrieval requests, such that the strengths of other recommendation paradigms (e.g., cross-domain recommendations of CF systems or customized requests of KB systems) can be leveraged for content-based approaches. The following chapter will describe in more detail how the characteristics of the LOD cloud can help to address some of the mentioned limitations of current recommendation methods.

Table 2.3: Strengths and weaknesses of recommendation paradigms

Paradigm	Strengths	Weaknesses
CF RS	<ul style="list-style-type: none"> • cross-domain recommendations • no knowledge base required • improve over time 	<ul style="list-style-type: none"> • data sparsity • new user problem • new item problem
CB RS	<ul style="list-style-type: none"> • no knowledge base required • improve with more preference data 	<ul style="list-style-type: none"> • new user problem • limited content analysis • overspecialization problem

Paradigm	Strengths	Weaknesses
KB RS	<ul style="list-style-type: none">• representation of non-product attributes• highly adaptive to user needs	<ul style="list-style-type: none">• high user effort required for the preference elicitation process• required knowledge engineering• often non-transferrable to other application domains
Hybrid RS	<ul style="list-style-type: none">• improvement of base learners	<ul style="list-style-type: none">• considerable tuning effort of recommendation models• costly in terms of required hard- and software resources

3 Linked Open Data for Recommendation Tasks

This chapter elaborates on the idea of applying Linked Open Data for recommendation tasks to address some of the previously outlined limitations of existing systems. It describes the origins of LOD publishing as well as the layers of the LOD technology stack (Sect. 3.1). Based on these descriptions, Section 3.2 explains the opportunities of Linked Open Data for recommendation retrieval, while Section 3.3 addresses the challenges that need to be handled by an RS when processing LOD.

3.1 Emergence and State of the Linked Open Data Cloud

3.1.1 Semantic Web

In 2001, Tim Berners-Lee, the father of the Internet, published a widely perceived article on the limitations of the World Wide Web at that time. He proclaimed that most of the then available content was designed to be consumed by humans. While the appropriate hard- and software for message interchange, page loading, and layout interpretation had been developed, the infrastructure primarily supported the consumption of documents. Only human beings could interpret the semantics of web data. However, the technology to derive meaning from web pages was missing then [34]. Whereas systems for semantic processing already existed in the early 2000s, these tools had the disadvantage of being designed for a particular task. They could only answer a limited amount of questions based on inference rules that had been defined for a certain domain. Interchangeability of data and rules across systems was not a given. For these reasons, Berners-Lee envisioned a decentralized web of semantic data, that could be accessed by different applications instead of serving application-specific tasks [34, 136]. The idea was to add technological layers of meaning, logic, and trust to the already existing infrastructure of the WWW (Fig. 3.1).

The vision was to turn the existing web into a platform that could answer questions by inferring information from known facts. Therefore, software agents would exchange semantic information without human intervention by making autonomous decisions based on available data [34]. The World Wide Web Consortium (W3C) adopted the proposal by Berners-Lee in the Semantic

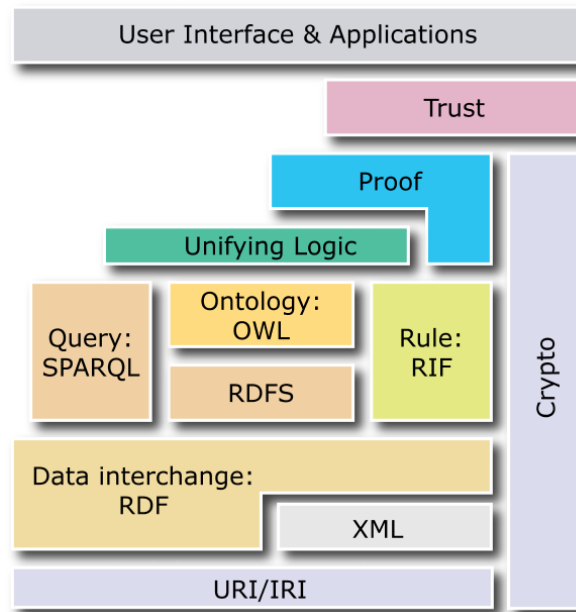


Fig. 3.1: Semantic Web technology stack [136]

Web technology stack [136]. One of the core building blocks of this conceptual framework was the idea to use unique identifiers (URIs and IRIs) to refer to objects, which can, in turn, be connected by the Resource Description Framework (RDF) vocabulary. RDF encodes facts as triple statements thus creating a decentralized information space. Subsection 3.1.3 will describe the exact syntax of RDF triples. However, to be able to infer meaning from RDF data, knowledge representation systems have to be implemented on top of it. They state which things exist in a domain and how they are related. These systems have long been known in artificial intelligence research under the term ontologies. An ontology language can express a domain model, from which facts can be derived that might not be explicitly stated [34]. Common ontology languages are the schema language for RDF (RDFS) and the Web Ontology Language (OWL) [136].

3.1.2 Linked Open Data

Over the past decade, the standardization activities of the W3C towards a Semantic Web infrastructure have led numerous organizations to provide their information sources in machine-readable formats. These sources stem from various domains and contain many facts about people, places, scientific publications, and multimedia or life science items [38]. While at first, advocates of an open web-based data space more frequently referred to the term Semantic Web, the expression Linked (Open) Data became more prevalent in recent years. However, the two words should not be confused. The Semantic Web vision describes a highly advanced state of the web, which enables intelligent software agents to autonomously derive and process meaningful information from openly accessible data sources. This vision requires a critical mass of open data and assumes that scalable reasoning technology already exists. While current reason-

ing systems have not yet achieved the latter requirement of web-scale performance, the former can be said to have evolved as the so-called Linked Open Data cloud [188].

After his first Semantic Web paper, Tim Berners-Lee refined his original statements regarding publication principles for data in an article published in 2006. He concretized his ideas by stating the following rules [32], which are nowadays often referred to as „Linked Data principles“ [38]:

- „Use URIs as names for things
- Use HTTP URIs so that people can look up those names
- When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL)
- Include links to other URIs, so that they can discover more things“

As envisioned by Semantic Web advocates, the web of data builds on the technology stack of the classical document web by relying on unique identifiers and the HTTP protocol for resource access.

Through the provision of open standards and active use of an existing decentralized infrastructure, the web of data shares many of the characteristics of the traditional WWW. Bizer et al. state the following principles for Linked Data publishing [38]:

- „The Web of Data is generic and can contain any type of data.
- Anyone can publish data to the Web of Data.
- Data publishers are not constrained in choice of vocabularies with which to represent data.
- Entities are connected by RDF links, creating a global data graph that spans data sources and enables the discovery of new data sources.“

While the above-listed principles follow the original idea of web-based publishing, Tim Berners-Lee later added a requirement that qualifies data for web-wide usage. Therefore, he made the crucial distinction between *Linked Data* and *Linked Open Data*. Datasets adhering to Linked Data standards (i.e., by being in RDF format and linking to other RDF datasets across the web) may not be openly accessible. Therefore, he declared that publishers should provide Linked Data under an open license [32].

The terms, under which a license is considered to be open, were specified by the Open Knowledge Foundation. Central to this definition is that it requires licenses to ensure usage free of charge and free redistribution and modification of data collections or any of their subsets. Additionally, the license must allow datasets to be used in conjunction with other content and

should guarantee that certain persons, groups or application areas are not excluded from using the data [176]. Licenses conforming to these regulations are the Creative Commons Attribution 4.0 license (CC-BY-4.0) or the Open Data Commons Attribution license (ODC-BY).

Having made this distinction between Linked Data and Linked Open Data, it must be stated that, to date, only a few openly accessible datasets possess licensing information [111]. Nevertheless, the expression Linked Open Data cloud is used to refer to the vast collection of interconnected resources freely provided (i.e., free of charge) on the Internet. Hence, the term Linked Open Data refers to a much broader category of open data publishing, i.e., to any dataset that adheres to „Linked Data principles“ and is accessible over the web [215]. Therefore, for the remainder of this thesis, whenever the term Linked Open Data is mentioned, it refers to the broader category.

The crawling activities of the LODStats project have revealed a tremendous growth of the LOD cloud during the last decade (see Figs. 3.2 - 3.4) [148, 151]. One of the first milestones of its evolution was the start of the DBpedia project in 2007. It publishes structured data from Wikipedia. Numerous data sets from diverse domains, such as geography, government, life sciences, linguistics, media, social media or publication have joined the DBpedia in the LOD cloud over the past ten years [148]. The key strength of this data web is that it connects resources through incoming and outgoing links across datasets [215]. Thus, a decentralized information infrastructure has emerged, which spans thousands of datasets that might be valuable for many data-driven applications [149], such as recommender systems.

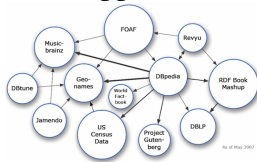


Fig. 3.2: LOD cloud 2007

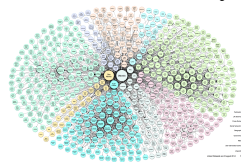


Fig. 3.3: LOD cloud 2014

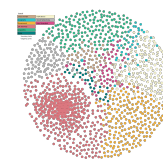


Fig. 3.4: LOD cloud 2017

3.1.3 RDF

The Resource Description Framework (RDF) is one of the central building blocks of the LOD cloud. Almost all openly available datasets structure the data with the help of the RDF vocabulary [215]. RDF is a language for expressing information about items, their interconnections and for defining (semantic) data models. The structured representation of information enables automatic processing by software applications. Items can represent documents, persons, physical objects or abstract concepts [216]. In an RDF graph, International Resource Identifiers (IRIs) uniquely describe most LOD resources. Hence, entities can be consistently referred to in a dataset [222]. Sometimes unique identification also implies the location and the means of access to this resource (e.g., as for „Uniform Resource Locators“(URLs)) in other cases, it does not. Therefore, URLs are a subcategory of IRIs. In Fig. 3.5, the DBpedia resource `dbr:Django_Unchained` representing the corresponding movie item is a URL, since the namespace `dbr:` indicates the location of the resource (i.e., `http://dbpedia.org/resource`). The IRI

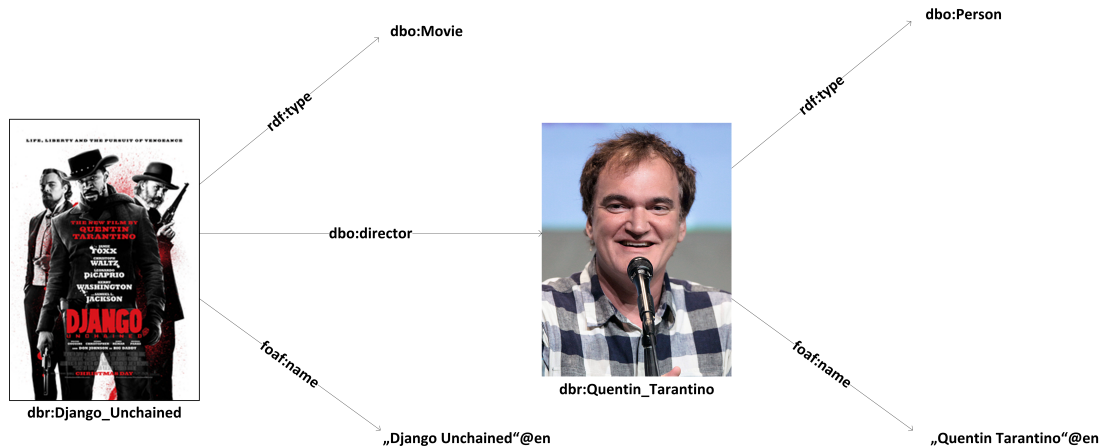


Fig. 3.5: Example RDF graph

specification can be found in the Requests for Comments (RFC) 3987 [77,216].¹

RDF graphs comprise triple statements in the form of <subject> <predicate> <object> assertions, where an IRI either occurs in the subject or the object of the triple statement. The RDF representation of the depicted graph of Figure 3.5 is given by the following statements:

```
dbr:Django_Unchained rdf:type dbo:Movie .
dbr:Django_Unchained dbo:director dbr:Quentin_Tarantino .
dbr:Django_Unchained foaf:name "Django Unchained"@en .
dbr:Quentin_Tarantino rdf:type dbo:Person .
dbr:Quentin_Tarantino foaf:name "Quentin_Tarantino"@en .
```

The example shows how RDF statements describe relationships between resources [216]. For instance, one of the triple statements declares that the movie „Django Unchained“ was directed by „Quentin Tarantino“. In this statement the resource `dbr:Django_Unchained` is the subject and the resource `dbr:Quentin_Tarantino` is the object of the triple. However, the latter LOD resource (`dbr:Quentin_Tarantino`) can also occur as a subject, e.g. in the triple that declares „Quentin Tarantino“ to be a person (`rdf:type`² `dbo:Person`³). The example triples illustrate how data graphs are set up with RDF. They also show how RDF syntax can be applied to describe and link entities. The structure enables identification of connections between items [216]. Aside from a subject and an object, triple statements comprise a predicate, which characterizes the nature of the relationship. The predicate is represented as a directed arrow (from subject to object) and has a label attached to it (e.g., `foaf:name`⁴) [216]. The link is often also called property. The subjects and objects constitute the nodes of an RDF graph, while the predicates represent directed edges [216]. In addition to IRIs, triples can also

¹Also note that the URI definition is a deprecated specification of unique identifiers. The IRI specification enhances the allowed character set of URIs [77].

² PREFIX `rdf:` <<http://www.w3.org/1999/02/22-rdf-syntax-ns#>>

³ PREFIX `dbo:` <<http://dbpedia.org/ontology/>>

⁴ PREFIX `foaf:` <<http://xmlns.com/foaf/0.1/>>

contain two other data types. One of them is blank nodes. Similarly to the application of IRIs, one utilizes blank nodes to reference a particular item. In contrast to IRIs, blank nodes are local identifiers that are only used in certain RDF stores [1]. They are applied when the data modeler is not able or not in the position to declare a global identifier. Blank nodes are syntactic placeholders marked by an underscore. For instance, the expression `_:b` is a blank node. The third category of RDF terms is literals. They describe an LOD resource with base type features [102, 222]. For instance, the movie title „Django Unchained“@en is a literal of the type string. Strings often have a language tag attached to them. Literals can have other base types, such as dates or numeric types (e.g., integer, double) [222]. IRIs, literals and blank nodes each represent distinct and distinguishable groups of RDF terms. Hence, they denote different types of data accordingly. For instance, the string literal „http://example.org“ is not the same as the corresponding IRI `http://example.org`.

The W3C specification of „RDF 1.1 Concepts and Abstract Syntax“ describes the notion of an RDF term formally [1]. A paraphrased version of this definition is shown below (Definition 1).

Definition 1 (RDF term). *An RDF term is a member of the set $I \cup L \cup B$, where I denotes the set of all IRIs, L the set of all RDF literals and B the set of all blank nodes.*

RDF triples consist of a subject, a predicate, and an object. They are formulated with the help of RDF terms. Different terms are eligible for each of the three parts. The following definition is taken from Perez et al. [185] (Definition 2).

Definition 2 (RDF triple). *An RDF triple consists of three components:*

- „the subject s , which is an IRI or a blank node ($s \in I \cup B$)
- the predicate p , which is an IRI ($p \in I$)
- the object, which is an IRI, a literal or a blank node ($o \in I \cup B \cup L$) “

Sets of RDF triples form RDF graphs. An RDF dataset (D) can be comprised of one to many RDF graphs. The respective definition of the RDF 1.1 syntax specification for RDF datasets is repeated below (Definition 3).

Definition 3 (RDF dataset). *„An RDF dataset (D) is a collection of RDF graphs, and comprises:*

- Exactly one default graph, being an RDF graph. The default graph does not have a name and MAY be empty.
- Zero or more named graphs. Each named graph is a pair consisting of an IRI or a blank node (the graph name), and an RDF graph. Graph names are unique within an RDF dataset.“

RDF graphs can be saved in different forms. There exist several RDF serialization formats, such as Turtle, JSON-LD, RDFa or NTriples. When the data processor has serialized the RDF dataset, it can be loaded into a triple store to serve as a LOD repository. Users and software agents can access it with standard RDF query languages and engines. The most prominent and widespread among these query languages is the SPARQL Protocol And RDF Query Language (SPARQL).

3.1.4 SPARQL

SPARQL is a W3C Recommendation and the standard for querying RDF triple stores [102]. Numerous LOD engines provide a SPARQL API [231]. The latest version of the SPARQL syntax specification is 1.1. It provides language constructs to formulate single as well as federated requests [102]. A SPARQL query can be one of the following four types:

- SELECT
- CONSTRUCT
- ASK
- DESCRIBE

They match the graph pattern specification of the query's `WHERE` clause against the RDF dataset. The result set differs for each kind of query. The `CONSTRUCT` and the `DESCRIBE` query type put out an RDF graph, while the `ASK` query simply returns a boolean value that indicates whether the processor could match the specified graph pattern with the triple statements in the LOD repository. `SELECT` requests represent the most important query form in the context of this thesis. SPARQL `SELECT` requests are useful for LOD-enabled retrieval tasks since they bind and return LOD resources to the variables occurring in query patterns. Hence, an LOD-enabled recommendation engine might profit from the ability to identify suitable LOD resources (i.e., recommendable items) [102, 222].

A `SELECT` query usually contains a prefix declaration, a `SELECT` statement and a `WHERE` clause. The prefix defines shortcut namespaces. For instance, in the following example, the namespace `dbr:` abbreviates the corresponding DBpedia IRI. Prefix specifications make SPARQL queries more human-readable while increasing formulation efficiency as they are valid for the entire query [102].

```
PREFIX dbr: <http://dbpedia.org/resource/>
```

The `PREFIX` declaration precedes the `SELECT` part of the request. In the `SELECT` part, users specify a subset of the variables that occur in the query pattern of the `WHERE` clause. They state, which bindings they want to obtain for each solution. In the `SELECT` as well as the `WHERE`

section of the query, variables are marked as character strings that start with a question mark (?) or a dollar sign (\$) [222].

The WHERE clause specifies the graph pattern (P) the processor should match with triple statements in the LOD repository. The most simple form of a graph pattern (P) is a triple pattern (t). A SELECT query with a single triple pattern looks as follows:

Listing 3.1: Example SPARQL query (simple matching)

```
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?movie
WHERE {
?movie dbo:director dbr:Quentin_Tarantino .
}
```

The query produces the result that is shown in Table 3.1, when issued against the RDF graph of the previous subsection (Subsect. 3.1.3).

Table 3.1: Solution set of a simple SPARQL query

?movie
dbr:Django_Unchained

The WHERE clause can also contain multiple triple patterns. An example for this is given in Listing 3.2. In this query, two conditions must hold to generate a match.

Listing 3.2: Example SPARQL query (multiple matchings)

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?movie ?person
WHERE {
?person rdf:type dbo:Person .
?person dbo:director dbr:Django_Unchained .
}
```

The query produces the following result (see Tab. 3.2).

Table 3.2: Solution set of a SPARQL query with multiple matchings

?movie
dbr:Quentin_Tarantino

Perez et al. describe the notion of a triple pattern formally. They use algebraic formalizations of core SPARQL elements [185]. The authors define the union of the sets of IRIs and literals ($IL = I \cup L$) and denote RDF terms as T ($T = I \cup B \cup L$). Additionally, they assume the presence of a set of variables, which is disjoint from IL and T . Given these preconditions, a triple pattern is defined (Definition 4).

Definition 4 (Triple pattern). *A triple pattern consists of a subject, a predicate and an object. IRIs, literals or variables are allowed in the subject and object positions of a triple pattern. The predicate position must contain an IRI or a variable (Eq. 3.1).*

$$t = (IL \cup V) \times (I \cup V) \times (IL \cup V) \quad (3.1)$$

When executing a SPARQL SELECT query, the engine looks for mappings of the query variables to RDF terms that are present in the dataset (D). According to Perez et al., a mapping μ from V to T can be defined as a partial function $\mu : V \rightarrow T$ [185].

Definition 5 (Evaluation of a triple pattern). *An evaluation of a triple pattern t over a dataset D is obtained by determining triple mappings ($\mu(t)$) and replacing all variables in the triple according to the mapping function (μ). The expression $var(t)$ denotes the set of variables that is present in t . The domain of the mapping ($dom(\mu)$) stands for the subset of variables V that occurs in μ (Eq. 3.2).*

$$[[t]]_D = \{\mu \mid dom(\mu) = var(t) \text{ and } \mu(t) \in D\} \quad (3.2)$$

The WHERE clause in a SPARQL query can also contain more advanced combinations of graph patterns. For instance, it is possible to join triple (or graph) patterns (e.g. as in Listing 3.2), to process additional SPARQL-based subqueries or to match optional graph patterns. The following overview of WHERE clause components refers to the graph pattern definition of the W3C SPARQL 1.1 syntax specification [102]:

- Conjunction of graph patterns (AND)
- Union of graph patterns (UNION)
- Optional graph patterns (OPTIONAL)
- SPARQL-based subqueries
- Negation of graph patterns (MINUS)
- Creation of values by expressions (BIND)

An example for the evaluation of a conjunction is shown in Equation 3.3.

$$[[(P_1 \text{ AND } P_2)]]_D = [[P_1]]_D \bowtie [[P_2]]_D \quad (3.3)$$

The conjunction of two graph patterns P_1 and P_2 is recursively defined as a join, provided that their respective mappings μ_1 and μ_2 are compatible. This is the case, when for all $x \in \text{dom}(\mu_1) \cap \text{dom}(\mu_2)$ the function returns the same value ($\mu_1(x) = \mu_2(x)$), such that the union of the mappings is also a mapping. It follows from this that two mappings with disjoint domains always fulfill the corresponding condition. Let Ω_1 and Ω_2 denote the two sets of mappings that have been obtained through the evaluation of two respective graph patterns over a dataset. The join of these mappings is defined by the union of compatible pairs of mappings [185] (see Eq. 3.4).

$$\Omega_1 \bowtie \Omega_2 = \{\mu_1 \cup \mu_2 \mid \mu_1 \in \Omega_1, \mu_2 \in \Omega_2 \text{ are compatible mappings}\} \quad (3.4)$$

Conjunction, as well as other types of advanced pattern matching, enables a wide variety of expressive queries. These query options are useful when obtaining data from RDF graphs. It is one of the reasons why the retrieval of information from LOD repositories bears excellent potential for recommendation tasks (Sect. 3.2.4). The following sections will describe the advantages of LOD in more detail.

3.2 Opportunities of Linked Open Data for Recommendation Tasks

3.2.1 Openness

As outlined before, limited access to high-quality data is one of the key problems of RS technologies. Users are often either reluctant to provide ratings or do not share enough preferences with other users in the database (*data sparsity*). A possible workaround for these problems would be to base recommendations on metadata information of potentially relevant items exclusively by applying a content-based approach. Thus, automatic suggestions could be decoupled from rating data. However, metadata might also be sparse or of low quality, such that the CB algorithm produces unfitting or biased recommendations.

The LOD cloud contains rich information sources from various domains. Recommendation engines can use these data sources without being hindered by cost, technological or legal barriers that might otherwise prevent access [176]. Since LOD datasets utilize standard protocols of the document web [119], technological openness is guaranteed. Any organization can publish and interlink data on the Internet in a machine-readable format. Additionally, the requirement to publish data under an open license ensures free of charge usage [32, 176]. While many datasets do not have a license, the majority of datasets still adheres to the beforementioned Linked Data publishing principles [215]. Hence, the LOD cloud in its current state can be described as an open data web. This characteristic might be especially helpful for organizations who do not

have the time or funds to maintain a high-quality database. Thus, even small online retailers might be able to offer helpful suggestions. Thereby, the application of Linked Open Data could potentially help to level the playing field in e-commerce.

3.2.2 Comprehensiveness

Over the last decade, the LOD cloud has grown into a giant interconnected knowledge graph containing billions of triple statements on a wide range of domains both within and across datasets [104]. One of the most prominent examples of comprehensiveness of a single dataset is probably DBpedia. It started as a grassroots movement that published data from Wikipedia in machine-readable format [22, 188] and is now one of the biggest cross-domain datasets in the LOD cloud [148]. The online encyclopedia Wikipedia is the largest reference site on the Internet. As of March 2017, the platform listed more than 70,000 active contributors and 41 million articles in more than 290 languages. Contributors write and edit entries collaboratively, such that chances are higher that articles cover a wide range of domains. Researchers found out that the crowd-based approach of the Wikipedia can compete with the centralized editing of the Encyclopedia Britannica. This finding was made in 2005 when Wikipedia was still in an early state and contained only 1% of the content it contains today [90]. Even if one assumes that the massive growth led to poorer article quality during the past years, the sheer quantity of information outperforms any centrally edited cross-domain reference source. It is this quantity and comprehensiveness that qualifies Wikipedia as a tremendously useful source for recommendation retrieval. Ten years ago, data from Wikipedia articles could only be retrieved by performing full-text searches and were thus hardly accessible for CB RS.

Now, it has become possible to query Wikipedia data in a database-like fashion because of the DBpedia project. The project extracts Wikipedia content by automatic identification of markup snippets from full texts, infoboxes, categories or geo-coordinates which are attached to the articles. The data is expressed in RDF [22, 144]. By these means, the project can keep up with the steady growth of Wikipedia. DBpedia has evolved into a large-scale knowledge graph comprised of structured multi-domain and multi-lingual data resources. The 2014 release contained more than 3 billion triple statements in 125 languages describing things from a wide range of domains. The English version holds machine-readable descriptions for 4.58 million items. Among them, the most frequently used item categories are persons, places, disease types and creative works [22].

Besides serving as a data mirror of Wikipedia articles, DBpedia is a central interlinking hub within the Linked Open Data cloud [144]. It is connected to all kinds of LOD datasets from various domains by more than 66 million incoming and outgoing links. However, DBpedia is not the only cross-domain dataset in the LOD cloud. Others are also based on Wikipedia (e.g., Freebase by Google [85] or YAGO by the Max Planck Institute for Computer Science [258]). Additionally, there exist general-purpose collections, such as the Integrated Authority File (GND)

of the German National Library, which is a registry for persons, organizations, events, topics or geographic information [88].

In addition to cross-domain datasets, there are all kinds of domain-specific collections in the LOD cloud, which could potentially serve various recommendation scenarios. In 2014, Schmachtenberg et al. identified eight significant domains as being most prevalent in the data web besides the cross-domain datasets [215]. Table 3.3 lists these domains and some example datasets accordingly.

Table 3.3: Overview of LOD domains in reference to [215]

Domain	Examples
Cross-Domain	DBPedia, Freebase, YAGO, GND
Geography	GeoNames, FAO geopolitical ontology
Government	Eurostat RDF dataset, World Bank Linked Data, data.gov.uk
Life Sciences	MeSH Thesaurus, Gene Ontology
Linguistics	WordNet, Lexvo
Media	New York Times LOD, DBTune
Publication	EconStor, AGRIS, data.bnf.fr
Social Networking	statusnet-quitter, statusnet-fragdev (GNU social)
User-generated	Revyu, flickr TM wrappr

For instance, in the geography domain, RDF collections characterize geographic entities by labels and geocoordinates (e.g., GeoNames) [88]. Others relate geopolitical terms to each other (e.g., the geopolitical ontology of the Food and Agriculture Organization of the United Nations [FAO]) [89].

Governmental datasets were published by federal or local authorities (e.g., data.gov.uk) or by intergovernmental organizations (e.g., the Eurostat RDF dataset or World Bank Linked Data). Often, they contain statistical information.

Life sciences collections typically describe biological entities from the subfields of medicine, zoology or biotechnology [215]. Prominent examples in this category are the Medical Subject Headings (MeSH). They serve as a thesaurus for indexing biomedical publications [46]. Another example from the life sciences domain is the Gene Ontology (GO). It is a knowledge representation system that was developed to describe how genes determine biological functions [87]

Other endeavors make linguistics resources publicly available. For instance, WordNet is the LOD version of an important lexical-semantic resource for the English language. It includes over 117.000 concepts [64]. Another undertaking of LOD publishing in the linguistics domain is Lexvo. This project offers word expressions in different languages and links them via semantic relations [145].

In datasets from the media domain, LOD resources describe movies, music, news-related entities or TV and radio programmes. A typical repository for news items is the New York Times dataset. The dbtune.org collection is a useful reference source for music information [215].

Bibliographic collections that contain publication metadata constitute another category of the LOD cloud. Often, they were produced by transferring library catalogs to RDF [188, 228]. The bibliographic dataset of the National Library of France (data.bnf.fr) is a prominent example in this category. By this means, the general public can benefit from nationally funded administration activities [2]. While library-based RDF descriptions often refer to printed documents, there are also collections that were directly derived from online document servers. For instance, the EconStor LOD dataset contains bibliographic information of the corresponding Open Access repository for papers in economics and business studies [79]. The LOD version of AGRIS is an example of a dataset, which is comprised of bibliographic descriptions for both printed and online material. AGRIS is one of the most extensive collections of references for agricultural publications. It contains bibliographic data for more than 5 million records. This data has been collaboratively generated by various agricultural information providers worldwide [12].

Crowd-based datasets were also gathered from open social networking sites, such as GNU-social (i.e., formerly known as statusnet). The Linked Data cloud contains several social microblogging graphs from servers running statusnet/GNU-social software [92].

Some RDF collections contain user-generated content. An example in this category is the LOD edition of the site revyu.com, where users can write reviews [203, 215]. Another representative dataset of user-generated content is „flickr™ wrappr“. It links DBpedia concepts to photos on Flickr [28].

The previous examples illustrated the topical diversity of the web. It covers a wide range of domains and subdomains which can potentially be applied to many information needs and recommendation requests.

3.2.3 Timeliness

The decentralized structure and the open technology stack of the LOD cloud are essential preconditions that enable publishers to update their data frequently. Providers might either synchronize collections with their local databases or directly publish new data. Just as the WWW architecture has accelerated the pace of document publishing, Linked Data principles provide a suitable framework for frequent update cycles of RDF repositories. This infrastructure facilitates the generation of relevant and up-to-date content-based recommendations from fresh data sources. The tremendous growth of the LOD cloud during the past decade might be an indicator that LOD collections are being frequently updated already. Nevertheless, there are two significant limitations to this assumption. One is that, as the LOD cloud evolves, data sources change and so do RDF links. Links, especially those between different datasets, can become outdated. Even though the architecture of the web is relatively tolerant to this, an overload of dead links can prevent unhindered data consumption [38]. Another limitation is that organizational surroundings and software tools for RDF conversion and web-based data publishing are not yet as advanced as the processes and software systems that help to generate web pages. Within

the WWW ecosystem there exist all kinds of tools, such as blog software or content management systems, which facilitate web-based online publishing. For LOD, on the other hand, each organization has its own approach and frameworks in place [12, 22, 144, 173]. Processes and software tools for LOD generation and maintenance are not yet standardized and thoroughly explored. Even though the LOD cloud provides the necessary preconditions for timely data updates, its full potential has yet to be realized.

Nevertheless, there already exists a software system which accelerates the process of automatically extracting and publishing RDF data on the web. It is called „DBpedia Live Extraction Framework“. It tackles the tension between the frequent update cycles of Wikipedia and the slow and laborious conversion to RDF. On average, Wikipedia articles get revised 3.3 million times per month. The framework enables live synchronizations with these edits. Therefore, it processes the changes of the Wikipedia update stream in an ad-hoc fashion and propagates these changes to a repository containing the latest DBpedia version. Through this piecemeal approach, the DBpedia is kept updated without imposing a substantial delay for changes [144, 169]. The opportunity to use machine-readable data snippets of recent Wikipedia articles has excellent potential for recommendation tasks. For instance, an online retailer could enhance information on the newest items with additional information from the knowledge repository. It is likely that the retailer can enrich his data with the most current Wikipedia information without having to tie up too many of his human resources for data maintenance tasks.

In summary, although LOD datasets do not by nature contain the most current information on any topic, the decentralized and convenient publication infrastructure of the data web enables frequent updates, which are a prerequisite for content-based recommendation retrieval on fresh data sources.

3.2.4 Data Expressiveness & Inference

The previous sections have shown that the LOD cloud has evolved into a massive knowledge graph. While classical relational databases organize data through tables, columns and foreign key relationships, graph-based RDF storage approaches are more flexible, since entities can link to one another in various ways. This also has implications for query processing. While a relational database has to perform numerous resource-intensive join operations to access multiple tables, link-based queries over graphs are often more efficient [44].

SPARQL is the standard query language for extracting data from RDF triple stores. As a graph-based language it is well suited to pose highly adaptive queries, which might be useful for RS tasks when users have specific information needs. For illustration purposes, consider the following example (Example 1):

Example 1. *Suppose an online shop customer is in search of female rappers from European countries. A relational database query for this request requires a profound knowledge of the*

underlying schema. Additionally, the product catalog of the retailer might be primarily based on a search engine index (e.g., as in Amazon’s product search [205]). In this case, it is likely that the request cannot be matched with the attribute-level metadata in the index. For the given query, this can be exemplified by looking at the usage constraint of requiring music acts to be European female rappers. It is more likely that the musician is associated with her country of origin, rather than with a particular continent. Thus, the query might not produce any search results. Hence, a SPARQL-based approach might be a useful alternative to improve search results for this information need. The corresponding query illustrates this example (Listing 3.3):

Listing 3.3: Expressive SPARQL query

```
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX dbc: <http://dbpedia.org/resource/Category:>
PREFIX dbp: <http://dbpedia.org/property/>

SELECT DISTINCT ?rapper
WHERE {
  ?rapper dct:subject dbc:Female_rappers .
  ?rapper dbp:origin ?subject .
  ?subject skos:broader* dbc:Europe .
}
```

The request matches not only directly linked item features, but also the surrounding graph structure of the item in question. The middle part of the last triple statement contains a property path (*skos:broader**) that marks a sequence of properties of arbitrary length. By these means, the query processor can retrieve different possible routes from one resource to another [218]. The property path does not require precise knowledge of the RDF graph but can match different versions of it. At the same time, the query defines a precise feature (i.e., *dbc:Female_rappers*) to identify suitable items.

The ability of SPARQL to express precise constraints as well as parts that invoke matchings of different triple patterns helps to query RDF graphs more extensively. In the given example, the property paths facilitate the generation of a more comprehensive result set than simple querying of item-level metadata. The LOD technology stack does not only provide options to query existing graph patterns but also enables inference of additional facts from available data sources by making use of ontologies that sit on top of RDF. For instance, RDFS is a light-weight ontology language which allows defining classes and subclass relations as well as class assignments. With RDFS it is also possible to declare domains and ranges for subjects and objects of a triple pattern [37]. By these means, RDFS ontology declarations can be used to infer additional triple statements. Thus, a more comprehensive RDF graph can be generated by obtaining its so-called RDFS closure. An example RDFS rule is shown in Equation 3.5. This example was taken from

a survey on LOD reasoning by Polleres et al. [190].

$$(?s, ?p2, ?o) \leftarrow (?p1, \text{rdfs:subPropertyOf}, ?p2), (?s, ?p1, ?o) \quad (3.5)$$

The antecedent of the rule declares that whenever a property $?p1$ occurs in a triple statement $(?s, ?p1, ?o)$ and it is a sub-property of another property $?p2$, the rule's consequent is that the super-property $?p2$ can also be assumed to be a property in the same triple statement.

Reasoning over such rules marks an interesting extension for query-based retrieval. For RS, rule engines or SPARQL query rewriting techniques could be applied to overcome issues of data incompleteness both in data repositories and in user queries. Example 2 illustrates this argument:

Example 2. *Suppose a user is going on a city trip to London. He has already skimmed through his tour guide in search of points of interest (POI) that might be worth a visit during the stay. Although the tour guide lists exciting locations, the user still would like to obtain far more suggestions for POIs that are not as frequently visited by other tourists. Therefore, he issues a SPARQL query against DBpedia asking for all places that are known to be located in London and are thus interesting enough to be mentioned by a Wikipedia article. However, by merely using the triples patterns shown in Listing 3.4, the user will not receive a single search result.*

Listing 3.4: SPARQL query for POI retrieval

```

PREFIX dul: <http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo: <http://dbpedia.org/ontology>

SELECT DISTINCT ?poi
WHERE {
  ?poi dul:hasLocation dbr:London.
  ?poi rdf:type dbo:Place .
}

```

This has to do with how DBpedia declares place-related triple statements. POIs in a city, region or country are usually not linked to the property `dul:hasLocation`, but to one of its sub-properties (e.g., `dbo:city` or `dbo:location`). By applying the RDFS rule from Equation 3.5) additional triple statements can be inferred. While the query of Listing 3.4 would not produce any search results, its rewritten version (Listing 3.5) returns many LOD resources when issued over DBpedia. Hence, the SPARQL engine would retrieve further LOD resources upon query rewriting, even though the RDFS closure of location-specific subproperty relations is not materialized in DBpedia.

Lightweight-reasoning in the context of query formulation may positively contribute to the quality of recommendation results. In concordance with the proposal of Polleres et al. to conduct LOD reasoning either on small RDF subgraphs (i.e., context-dependent reasoning) or

Listing 3.5: SPARQL query after RDFS query rewriting

```
PREFIX dul: <http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology>

SELECT DISTINCT ?poi
WHERE {
  ?poi ?p dbr:London .
  ?p rdfs:subPropertyOf dul:hasLocation .
  ?poi rdf:type dbo:Place .
}
```

solely with regard to a particular vocabulary/set of rules (i.e., authoritative reasoning), it is assumed that tailoring inference rules to a specific request can help to address challenges of scalability and vocabulary inconsistencies [190].

In this context, it is hypothesized that inference through SPARQL querying is better suited for LOD-enabled RS than materializing triple statements as the strengths of fresh data sources (Subsect. 3.2.3) would be undermined by the latter approach.

3.3 Challenges of Linked Open Data for Recommendation Tasks

3.3.1 Data Quantity

Even though the LOD cloud provides interesting opportunities for recommendation tasks, it also has some shortcomings. A system residing on top of the LOD technology stack needs to address these issues as well. In Subsect. 3.2.2, it was explained how the vastness of machine-readable data could be of value for recommendation retrieval. However, this abundance of data also has a downside: The more triple statements are available, the more the recommendation engine needs to process. Ziegler anticipated this problem even before the growth of the Linked Data cloud took place. In his 2004 article „Semantic Web Recommender Systems“ he explains that centralized systems have the advantage that computational costs can be estimated based on the number of items/users in the local database [261]. They allow for restrictions on community and item sizes when identifying neighborhoods. Such limits are critical regarding performance since similarity calculation is usually computationally expensive. He correctly points out that a data cloud containing millions of items can only be efficiently processed once there are suitable filtering mechanisms available to reduce data throughput [261].

During the last decade and as anticipated by Ziegler, the LOD cloud has reached an extent that cannot be easily handled by available recommendation algorithms. Research projects have revealed a tremendous growth of the LOD cloud. For instance, in their 2014 report on the adoption of Linked Data best practices, researchers of the „LOD cloud diagram“ project stated

that the data web at that time consisted of more than 8 million resources [215]. The LODStats project, on the other hand, has found more than 54 million entities from its recent data crawl in 2016. It remains unclear, both whether the terms „resource“ and „entity“ refer to the same type of IRI declaration and whether these numbers were derived from analyzing the same datasets. However, it can still be assumed that the LOD cloud has considerably grown since its beginnings in 2007. The number of available datasets backs this assumption. From 2007 to 2017 the amount of LOD collections has risen from a total of 12 datasets in 2007 to 1.146 datasets in 2017 with an average growth rate of 48,5% [148].

While the number of resources and datasets serves as an indicator for the state of the LOD cloud, this information can bias quantity estimation, given the fact that a single IRI can occur multiple times within and across datasets and that dataset sizes can vary tremendously. Therefore, the number of triple statements in the current data cloud is important as well. For this purpose, the LODStats project determined some figures. In 2016, within the course of their latest data crawl, the researchers identified more than 290 billion triple statements [148]. Whereas some of these statements might as well be duplicates, this number is still impressive when considering LOD-enhancement for RS.

Against the background of the data quantity in the LOD cloud, a LOD-enabled system would have to identify suitable strategies to handle data throughput efficiently.

3.3.2 Data Heterogeneity

A LOD-enabled recommendation engine will also have to tackle the heterogeneity of RDF resources. For instance, item descriptions can contain diverse elements which need to be treated differently by the engine to calculate item-to-item similarity values [1]. LDRS researchers have proposed methods for handling common data types in the LOD cloud. However, these methods imply additional challenges. For instance, while IR metrics can easily consume IRIs [66, 67, 165, 186], string literals need to undergo natural language processing (NLP) operations, such as tokenization, stemming and character removal before any similarity metric can be applied [206]. In addition to well-structured IRI resources and text data, numerical values can also be part of item descriptions. The choice of an appropriate similarity metric and corresponding preprocessing operations depends on the nature of the particular item feature [186]. As such, the LOD-enabled engine would have to handle data type diversity. However, this can cause problems in the workflow as item features might require different processing steps and individual similarity values need to be aggregated accordingly.

The second problem regarding LOD heterogeneity concerns the diversity of metadata models, which can be attributed to the web of data's history of origin. The LOD project started as a grassroots movement of research labs and organizations. Being decentralized and having been collaboratively built by different parties, the LOD cloud offers a wide variety of data models [38]. While RDF is the central building block of the LOD cloud with more than 98% of

datasets using it, it does not contain any domain-specific terms [215]. Additional vocabularies need to provide them instead. The LOD community strongly encourages usage of well-known terminologies, such as the Friend of a Friend Ontology (FOAF), the Simple Knowledge Organization System (SKOS) vocabulary or the set of expressions of the Dublin Core Metadata Initiative (DCMI) [104]. These vocabularies are among the ten most widely used terminology sets within the LOD cloud [215].

Even though some vocabularies have become de-facto standards, there still exist numerous datasets that apply different vocabularies or proprietary terms to describe items [38]. Within the course of their latest web crawl, the researchers of the LODStats project identified 2593 different vocabulary namespaces in the LOD cloud [150]. This is a tremendous amount of vocabularies to consider when intending to develop a generic recommendation framework that can process sources from different LOD datasets and repositories. Item features have to be selected and processed to generate CB suggestions. Because of the various vocabularies in place, feature data might have been declared according to diverging conventions, e.g., through different properties. For instance, the LOD project Bio2RDF, which provides bibliographic metadata for publications from the life sciences domain, utilizes the property `dct:title` to describe a resource's title [36]. In the LOD repository EconStor, on the other hand, a title is either declared with the help of the property `rdfs:label` or with the property `dc:title`. This small example alone illustrates that datasets may use various vocabularies for the same entities and that different namespaces can describe semantically similar item features. Additionally, the LOD cloud contains resources that describe real-world entities from a wide variety of topics. As a result, typical item features and the range of allowed data types for these features will vary for different item types. For instance, according to the DBpedia ontology, a person (i.e., `dbo:Person`) has a completely different set of attributes than a movie resource (i.e., `dbo:Film`). Whereas a person should have a `dbo:birthDate` or a `dbo:birthPlace`, a movie will be characterized by its director or actors that are associated through properties, such as `dbo:director` or `dbo:starring` [61, 138].

Due to the heterogeneity of the LOD cloud, a generic approach of similarity calculation will have to be identified to be able to execute recommendation workflows for different datasets and target domains.

3.3.3 Data Distribution

The challenge of distributed LOD collections is related to the issue of data heterogeneity. Since the data web resides in over 2973 datasets which can either be downloaded or accessed via SPARQL endpoints, data sources are spread over numerous datasets and repositories [149]. While it is a strength of the LOD technology stack that it enables decentral publication of data, this imposes challenges for data processing. For an engine to produce automatic suggestions based on decentralized repositories, triple statements should be unified during the process of

recommendation retrieval. One solution would be to collect, preprocess, index and store the data into a central triple store. This materialization-based approach facilitates fast query response times at the expense of data freshness. However, it leads to high preprocessing workloads and undermines the original vision of a web that can be accessed on-the-fly [240].

Therefore, in the context of SPARQL querying, there exist other approaches that facilitate multi-dataset access. One approach is that of client-side virtual data integration, where heterogeneous data sources are translated into an integrated local target representation. Client-side data integration is especially suited for distributed datasets that contain many proprietary terms or IRI aliases. Differing terms among datasets are matched and replaced with an unambiguous target expression in the integrated local schema, such that resources can be queried without explicit knowledge of the different underlying schemas. By this means, aspects of data translation and identity resolution can be addressed. The most prominent example of virtual LOD integration is the Linked Data Integration Framework (LDIF). The web-based systems group at the Freie Universität Berlin developed this framework during a research project [217]. It applies the R2R Mapping Language to translate different terms into a unified target representation. However, the generation of LOD mappings and subsequent integration in a local schema is cumbersome and can only be partly automatized [39].

It might not be necessary to maintain a local unified representation for certain data sets and application scenarios. Rather, queries can be executed by solely relying on the individual schemas that are present within the distributed repositories. Currently, the LOD cloud contains billions of triple statements from various domains. As previously mentioned, there will be cases where the same IRI identifiers are found in different datasets. Additionally, at least in some domains, the web of data is highly connected through cross-dataset relations (e.g., `owl:sameAs`, `skos:exactMatch`), which are often referred to as entity resolution links [215,217]. Cross-dataset links and matching IRIs can be processed to query remote repositories on-the-fly without requiring any preprocessing operations. These so-called distributed query processing (DQP) approaches decompose a request into its subparts and issue them against the respective repositories [240].

Distributed SPARQL querying can be categorized into two groups, namely link traversal and federated systems. Link traversal approaches are based on IRI access and runtime retrieval [200]. Link traversal systems profit from key Linked Data design principles, where IRI identifiers should be dereferenceable through the HTTP protocol, such that an HTTP GET operation provides useful RDF descriptions and additional links to other IRIs. By this means, an agent can navigate from entity to entity [32]. LOD traversal shares features of common web crawling, since it is facilitated by the same technologies. One central advantage of this access method is that it can retrieve the most up-to-date resources currently available in the LOD cloud. Nevertheless, this comes at the cost of response times, since the dereferencing of highly connected IRIs might lead to an overload of resource accesses. Additionally, in scenarios that involve loosely connected resources, dead ends will be reached too soon, even though there still exist

more resources that would fit a query [240].

The other category of DQP systems retrieves pre-indexed data from remote triple stores through sending SPARQL queries to different endpoints, instead of traversing LOD resources [198,200]. Federated querying usually diminishes response times, but may sometimes come at the cost of slightly outdated data sources. However, as of 2017, the SPARQL endpoint infrastructure is quite advanced. Even though only a small proportion of linked datasets is accessible via remote RDF stores [215], the vast majority of triple statements can be retrieved from SPARQL endpoints [149]. This may seem contradictory at first, but the assumption that the largest and therefore most prominent datasets are published through endpoints, helps to gain an understanding of the nature of the current data web as a rich information space that can be easily accessed through Application Programming Interfaces (API) for RDF data.

Therefore, of all the presented data distribution methods, the approach of federated querying seems to be the most promising in terms of computational cost and feasibility [200]. The importance of this approach is further underlined by the fact that the W3C has issued a recommendation for a SPARQL 1.1 federation syntax [191].

Bearing the prominence of federated querying in mind, it must be noted that these approaches can only execute SPARQL queries over triple stores. What still remains unclear, however, is how federated data repositories can be integrated into the context of recommendation retrieval. Clearly, there are already effective approaches for extraction and unification of LOD resources in place, but suitable methods for integrated similarity calculation and resource ranking still have to be identified.

4 Requirements Analysis

This chapter presents a requirements specification that is based on the limitations of existing approaches to recommendation retrieval as well as on the characteristics of Linked Open Data. It will elaborate on the determining factors of developing a specification for a LOD-enabled RS (Sect. 4.1). It defines functionalities that are aimed at making the most of LOD for recommendation tasks, while at the same time addressing the challenges of RDF data (Sects. 4.2 and 4.3). The requirements specification also takes into account qualitative performance aspects since the to-be-developed engine should improve and enhance existing approaches to recommendation retrieval (Sect. 4.4).

4.1 Characteristics of a Requirements Analysis for a Linked Open Data-enabled Recommender System

The central objective of this thesis is to develop a recommendation engine that addresses the strengths, weaknesses and technological characteristics of the LOD cloud. Standard software engineering methods are applied to ensure that the to-be-developed system is fit for purpose. The systematic approach is chosen to minimize the risk of designing an engine with useless functionalities. The previously identified opportunities and challenges of LOD processing will guide the requirements elicitation process, thereby making sure that the final system reflects the characteristics of Linked Open Data as well as the requirements of recommendation tasks and potential stakeholders [118].

The term stakeholders refers to all persons who influence the system specification. It includes individuals that use, operate or run a system (e.g., end users, administrators, developers or software testers) [209]. In the requirements elicitation process at hand, however, aspects of system maintenance will not be addressed because they do not reflect the main purpose of this thesis. The analysis is carried out by considering potential system users at the back- or front end of the LOD-enabled RS, instead. An end user may be a visitor or customer of an online platform (e.g., an online shop or a web-based information platform). A backend user is an administrator that adapts recommendation workflows or retrieval patterns to recommendation scenarios. Therefore, besides addressing all technical aspects of LOD, the set of requirements should contain specifications targeting the ease of retrieval as well as the nature of suggestions, since the engine should provide an added-value regarding recommendation generation and quality.

Before implementation, general goals have to be broken down into specific requirements that

address certain characteristics of the recommendation engine. A requirement is a feature or ability which is needed to solve a problem or to reach a goal. It should be unambiguously formulated in written form such that requirement fulfillment can be checked at a later point of the development process. This verification is necessary to ensure that the recommendation engine meets the specified demands [118]. For some requirements, a prototypical implementation will suffice to demonstrate the feasibility of the feature, whereas for others a more profound analysis needs to be conducted (e.g., measurement of computational times) [209].

According to the Software Requirements Specification (SRS) of the Institute of Electrical and Electronic Engineers (IEEE), requirements can be divided into the following categories: functional, performance, interface, design, process and non-functional requirements. Of these categories, the ones referring to interface, design, and processes will not be looked at, as they concern minor aspects in the context of the software engineering process at hand. Since the main focus of this thesis is to develop a system that can improve existing approaches through LOD usage, interface and design requirements should be the focus at a later stage of the implementation. Requirements in the process category will not be looked at because they refer to the conduction of the software engineering process (e.g., by demanding that the process complies with national legislation) [118].

This requirements analysis will focus on functional, performance and non-functional requirements since they reflect the research agenda of this thesis. Functional requirements refer to all specifications that state the tasks the system needs to perform. From a quantitative perspective, performance requirements specify the speed or effectiveness of certain operations that must be guaranteed by the system. Non-functional requirements refer to all specifications that are related to quality aspects or human factors. By defining requirements in this category rather than specifying what should be done, it is stated how the engine should perform its tasks. Features can be expressed by using characteristic terms, such as flexibility, portability or user satisfaction [118]. In the following subsections, requirements for each category will be defined and described in detail.

4.2 Functional Requirements

4.2.1 Requirements to Similarity Detection

Sections 3.2 and 3.3 have outlined strengths and weaknesses of LOD for recommendation tasks. Suitable approaches to similar resource retrieval should be identified to take advantage of these characteristics for content-based suggestions. CB systems process text data and thus resemble information retrieval (IR) systems. In most standard IR systems, however, query results are generated from a closed document collection. The engine usually indexes these documents before live application to facilitate fast responses [158]. Since fast processing is a desirable feature in any retrieval context, the engine should be able to generate recommendations quickly.

However, since resources are not contained in closed collections, but are available from open repositories, other retrieval strategies have to be identified. Pre-indexing is time-consuming, and it might undermine the timeliness aspect of LOD. Therefore, the system should perform ad-hoc similarity calculations. Another reason for this requirement is that the LOD infrastructure should be remotely accessible by small application providers since they might not be able to afford the human and technical resources to set up and maintain an index of several LOD repositories. Hence, RS providers should be able to retrieve recommendations for an input user profile through an API. The ability to generate *on-the-fly recommendations* has another advantage: While traditional retrieval systems mostly deal with attribute-level metadata [93], the graph structure facilitates expressive filter conditions which cannot be as easily indexed. It is thus more promising to combine similarity calculation with graph-based constraints in an ad-hoc fashion to allow for flexible customizations at runtime.

Requirement 1 (RQ1: On-the-fly similarity calculation from LOD repositories). *The system shall retrieve similar items to a profile on-the-fly from a LOD repository. A user profile can contain one to many preferences for items (LOD resources), which are uniquely identified by an IRI.*

The vocabulary heterogeneity of the LOD cloud (Subsect. 3.3.2) hinders broad applicability of recommendation algorithms. For the similarity calculation to be executable over different collections, metadata descriptions should conform to the same data model. Since some LOD vocabularies have become de-facto standards by now, they should be applied to overcome problems of data heterogeneity.

Requirement 2 (RQ2: Applicability to numerous LOD datasets (breadth of retrieval possibilities)). *The recommendation engine shall be able to perform similarity calculation for a significant amount of LOD datasets and a wide range of domains. Similar resource retrieval shall rely on de-facto standard LOD vocabularies and shall not be impeded by the proprietary specificities present in datasets.*

Besides the aspect of broad applicability, the similarity detection procedure should also be able to use available knowledge sources more extensively. Since the framework is required to be executable over various datasets by relying on a few vocabularies, some form of information loss will have to be tolerated. Therefore, despite its generic applicability, similarity calculation should be adaptable such that users can browse collections from different angles. The intention is to develop a method that can re-rank resources according to various user inputs concerning the depth of similarity analysis. This requirement aims at enabling users to browse knowledge graphs more comprehensively.

Requirement 3 (RQ3: Flexible similarity detection - Applicability to different granularity levels of similarity (depth of retrieval possibilities)). *The system shall provide recommendations*

according to different item similarity thresholds. Therefore, resources shall be re-ranked, whenever specifications regarding similarity values change.

4.2.2 Requirements to Linked Open Data Integration

Despite aspects regarding similarity calculation, the technological infrastructure of the LOD cloud has to be taken into account as well. In Subsection 3.3.3, it was discussed that the distribution of RDF collections represents an obstacle for generating recommendations that accurately reflect the current state of the data web. In this context, two different strategies of distributed processing were introduced, namely federated endpoint querying and link traversal. It is hypothesized that the points made in favor of SPARQL endpoint queries over link traversal strategies can be transferred to the context of LOD-enabled RS as well. Thus, the ability to perform SPARQL query processing over potentially remote and federated endpoints is a vital prerequisite for *on-the-fly recommendations* because it creates convenient retrieval options for RS administrators and users.

Requirement 4 (RQ4: SPARQL endpoint integration). *The system shall provide an interface to openly accessible SPARQL endpoints. It shall retrieve, extract and process data from these endpoints to identify similar items for a given user profile.*

The SPARQL API requirement only refers to the aspect of similarity calculation. In addition to that, it is equally important to specify a system component that integrates user filters into the process of recommendation retrieval. Since the expressivity of the RDF data model bears excellent potential for personalizing requests, a LOD-enabled RS should provide a feature that facilitates the formulation of powerful user queries as well. For this purpose, user constraints should be expressible via graph patterns that the engine then maps to RDF resources. A query language that utilizes SPARQL 1.1 syntax elements could meet this demand. SPARQL applies graph pattern matching and is thus suited for expression of highly individual filter conditions [192]. By this means, user constraints can be combined with similarity calculation during the process of metadata extraction from SPARQL endpoints. The provision of a query language requires processing units (i.e., a parser and a compiler) which verify the syntactic and semantic correctness of queries and process the SPARQL-like language constructs. By these means, RS administrators and users can formulate recommendation requests that can be answered right away.

Requirement 5 (RQ5: Query language and processing units to formulate and execute recommendation requests with SPARQL syntax elements). *The recommendation engine shall provide a language that can express SPARQL-based queries such that recommendation requests are executable over openly accessible SPARQL endpoints. The language shall provide constructs to formulate preferences and filter conditions (i.e., constraint-based retrieval). In conjunction*

with the query language, the system shall provide a parser and a compiler to process recommendation requests correctly.

While Requirement 5 requests that the language is able to formulate recommendation queries, it does not define the extent of its expressiveness. Even though a pre-selection of items through SPARQL filters may already enable advanced requests, retrieval possibilities are not thoroughly exploited by compliance to this specification. Instead, queries over RDF graphs might be even better suited for retrieval, when an engine can combine workflow steps of graph-pattern matching and similarity calculation at different stages of the retrieval process. SPARQL-like filters should be applicable for user profile generation and pre- and post-selection of LOD resources. Depending on the workflow stage, reduced item sets could be either fed into the similarity engine (*prefiltering*) or joined with similar items past to the process of similarity calculation (*postfiltering*). The possibility to flexibly combine these techniques facilitates *advanced recommendation requests* that are not yet expressible with existing LOD-enabled RS. Naturally, both the parser and the compiler of the query language need to be able to process advanced queries as well.

Requirement 6 (RQ6: Ability of the query language to express advanced recommendation requests). *The query syntax shall contain language constructs such that advanced recommendation requests can be formulated. The term „advanced“ refers to the ability of the language to define combinations of user constraints and similarity calculation at different stages of the retrieval process. Hence, the language shall contain constructs that apply SPARQL-like filter patterns for profile generation, pre-selection of metadata descriptions or postfiltering of similar items.*

4.2.3 Requirements to Virtual Data Integration

Challenges of LOD-enabled retrieval concerning the distributed nature of the web of data have already been outlined in Subsection 3.3.3. Therefore, the system should provide retrieval techniques that generate *cross-repository recommendations*. Data integration strategies affect different features of the system. One is the query language, which needs to provide syntax elements that trigger distributed resource retrieval. Additionally, the recommendation engine should facilitate adaptation of recommendation requests to the characteristics of other datasets than the default dataset. This requirement assumes that each running instance of the to-be-developed system is configured to generate recommendations from a specific SPARQL endpoint containing a major RDF dataset (e.g., DBpedia). By these means, data models and vocabulary specificities are familiar to the user (e.g., to the system administrator). Additional LOD resources should be extractable from further SPARQL endpoints which provide access to related datasets. Hence, it becomes possible to generate recommendations from other repositories to supplement results whenever this is necessary. The engine should generate *cross-repository recommendations* through basic techniques in which a user profile and potential filters guide the retrieval

process. More advanced retrieval patterns, such as distributed *preference queries* or *postfilter requests* (Requirement 6) are not covered by the specification at hand because they would, in turn, require advanced data integration techniques. Since the main focus of this thesis is the generation of customizable LOD-enabled *on-the-fly recommendations*, *cross-repository suggestions* are a subtopic, for which only preliminary investigations will be conducted. However, even simple *cross-repository requests* require adjustments in terms of query language constructs (Requirement 7) as well as similarity calculation (Requirement 8).

Requirement 7 (RQ7: Ability of the query language to formulate cross-repository requests). *The query syntax shall provide language constructs to express cross-repository queries. In these queries, the user profile refers to items from the default source repository, based on which the retrieval of similar items from a specified target repository is triggered.*

Requirement 8 (RQ8: Ability of the recommendation engine to process cross-repository requests). *Upon receiving a parsed cross-repository query, the engine shall direct requests for metadata descriptions to the specified target SPARQL endpoint and generate recommendations based on this data.*

4.3 Performance Requirements

In any recommender system, computational performance is critical. While for CF systems, the number of required operations is often reduced by low profile overlap among users, item matrices in CB systems are usually not sparse. Hence, the engine has to determine item-to-item similarities for almost every item pair, as live systems need to provide fast response times. Therefore, similarity values are often calculated offline to be quickly available at runtime [152]. However, this undermines ad-hoc queries since filter constraints have to be known beforehand. In these cases, the recommendation model is „hard-wired“ into the system and users cannot customize their requests [250]. However, as user requirements cannot be foreseen, recommendation models should be adaptable at runtime as was specified in Section 4.2 .

Hence, efficient and scalable processing strategies have to be identified. Query-based RS may not operate as fast as RS, which utilize pre-computed similarity values. However, the engine’s response times should still be reasonable. They do not only depend on the system, but also on the network load and the performance of the (remote) SPARQL endpoints. Assuming that these factors may vary, the actual time that is required to generate recommendations should be kept as short as possible. Regardless of the external aspects, the workload of the engine is subject to the size of the user profile and the size of the queried datasets. User profiles can contain many items, based on which the engine assembles the final result list. Therefore, response times for a separate single-item request should be on the scale of seconds to keep processing times within a manageable time span.

Requirement 9 (RQ9): Ability of the recommendation engine to provide quick responses to single-item on-the-fly recommendation requests). *The recommendation engine shall answer single-item requests on average within a couple of milliseconds. Additionally, response times shall not be longer than a few seconds for queries that involve larger datasets.*

Given the fast growth of the LOD cloud during the last decade (see Sect. 3.2.2), the recommendation engine should be scalable to increasing data sources. In this context, response times should not grow exponentially when new RDF triples are added to a dataset. Instead, the system is expected to adapt reasonably to changes, preferably with linear growth of response times.

Requirement 10 (RQ10): Ability of the recommendation engine to handle large datasets well). *Response times of the recommendation engine shall increase reasonably for a growing number of triple statements. The exponential growth of processing periods shall be avoided.*

4.4 Non-functional Requirements

The IEEE Software Requirements Specification divides non-functional requirements into quality requirements and human factor requirements. Quality requirements refer to typical system characteristics, such as portability or reusability. These requirements should already be taken into account when specifying functional requirements. Human factor requirements refer to the perceived outcome of a user's interaction with the system. In general, this concerns abstract concepts, such as safety, maintainability or *user satisfaction* [118].

The requirements analysis at hand primarily focuses on the aspect of *user satisfaction*, because the key goal of this thesis is to develop an engine, which can process LOD data sources in such a way, that suggestions have an added-value to the user. Therefore, the focal point of the analysis is to invent techniques that improve common recommendation strategies.

User satisfaction can refer to different dimensions, such as to algorithmic performance or to the usability of the interface [193]. However, in the course of this thesis, the term *satisfaction* directly concerns the quality of recommendation lists (see Subsect. 7.1.4). Hence, other dimensions, such as the user interface, are left out of the analysis.

The previous sections have introduced system features that refer to novel recommendation strategies. These strategies should improve the quality of conventional content-based recommendation approaches. For instance, Requirement 3 addresses the aspect of *flexible similarity detection*, which focuses on enhancing possibilities of knowledge graph exploration. Therefore, recommendations resulting from this approach should provide an added value to users.

Requirement 11 (RQ11): Ability of the recommendation engine to improve recommendation quality through flexible similarity detection methods). *The flexible similarity detection feature of the system shall provide better recommendations, than a regular content-based strategy.*

Requirement 5 specifies that SPARQL syntax elements shall be combinable with similarity calculation. In consequence, the application of SPARQL query constraints (i.e., *constraint-based retrieval*) should lead to recommendation lists of higher overall quality.

Requirement 12 (RQ12: Ability of the recommendation engine to improve recommendation quality through SPARQL query constraints). *The possibility to formulate SPARQL query constraints shall provide an added-value to users. Therefore, filtered results shall be of higher quality, than non-filter recommendations.*

One of the key arguments for executing SPARQL query elements in conjunction with similar resource retrieval is that the expressiveness of RDF knowledge graphs may enable powerful recommendation strategies. While filtering options have already been successfully implemented in faceted search systems, they mostly only match the direct attributes of an item [93]. However, it is assumed that RDF-based filters provide an added-value since they can be formulated as *expressive* graph constraints. By this means, query rewriting and lightweight reasoning can be applied to infer additional facts from LOD repositories (see Subsect. 3.2.4).

Requirement 13 (RQ13: Ability of the recommendation engine to improve constraint-based recommendation retrieval through expressive SPARQL queries). *The possibility to formulate expressive queries shall improve recommendation results. The term „expressive“ refers to filters that go beyond simple attribute-level metadata descriptions by including the surrounding RDF graph structure of an item. Hence, for constraint-based queries, the engine shall achieve better recommendation results when applying expressive instead of simple user filters.*

Requirement 6 specifies that the system should facilitate the formulation of advanced queries by enabling users to apply filters at different stages of the recommendation workflow. It is assumed that *advanced query patterns* can sometimes help to grasp a user's information need better than a simple recommendation request. According to Manning et al. the term information need refers to a topic that is of interest to a user and is different from the actual query. The query is the tool with which a user conveys what his/her interests are [158]. Hence, enhanced query options might equip users with more possibilities to accurately state their preferences. Additionally, *advanced query patterns* might not only be useful regarding information need translation but could also enhance LOD retrieval possibilities to explore RDF graphs better. Both the aspects of improved LOD navigation and information need expression are assumed to increase recommendation quality. Therefore, the engine should be able to enhance *user satisfaction* through *advanced retrieval* options.

Requirement 14 (RQ14: Ability of the recommendation engine to improve recommendation quality through application of advanced query patterns). *The recommendation engine shall improve the quality of recommendation lists through the application of advanced query patterns for application contexts, where either user requirements or the nature of the LOD cloud demand enhanced retrieval possibilities.*

5 Related Work

Based on the requirements specification for a LOD-enabled RS that was detailed in Chapter 4, this chapter presents the results of a literature survey on approaches in the fields of IR, recommender and Linked Data-enabled retrieval systems. The review takes into account the functional requirements of the specification and determines whether they can be found in existing non-Linked Data (Sect. 5.1), or Linked-Data enabled engines (Sect. 5.2). The chapter concludes with a summary of the most important findings and identifies the research gaps that should be addressed by the to-be-developed recommendation engine (Sect. 5.3).

5.1 Non-Linked Data Systems

5.1.1 Search Systems

Researchers have been investigating suitable approaches to process advanced search requests over databases for many decades. In the context of computer-aided retrieval, early systems adopted a strategy that is often classified under the term *parametric search* [239]. This technique relies on bibliographic metadata and can still be found in many online public access catalogs (OPACs) of libraries [13, 25, 71].

The screenshot shows the search interface of the Thuringian University and Regional Library (ThULB) Jena. The main search area is titled "search filter" and contains a search input field with the instruction "Fill in one or more words in the search form below and add the desired settings". Below the search field are four dropdown menus for search criteria: "[ALL] all words", "[TPC] all topics", "[PER] person/author", and "[TIT] title (keywords)". Each dropdown menu is followed by an "and" dropdown menu. Below these are filter options: "sort by" (set to "year of publication"), "year of publication" (with a range input field and example "1948-1980 or 1948- or 1955"), "language" (set to "-- All languages --"), and "country" (set to "-- All countries --"). There is also an "approximate search" checkbox and a "material selection" section with "all" and "none" options.

Fig. 5.1: Online public access catalog (OPAC) of the Thuringian University and Regional Library (ThULB) Jena

A *parametric search* interface provides retrieval capabilities with which users can restrict search

results to specific aspects (e.g., author, subject or year of publication) by using Boolean operators (see Fig.5.1). Hence, the engine matches search terms that are specified by the user against the metadata values of the document collection and combines result sets according to the specified logical AND/OR operators of the query. However, leaving the constraint selection process entirely up to the user may impede efficient retrieval because it is likely that users either do not know which values to define or they might specify constraint combinations that lead to a dead end during a search. That is why researchers have proposed utilizing faceted navigation, where possible values for each facet are shown in the interface and thus help users to select only constraints that do not lead to empty result sets [239].

The strategies of *parametric search* and *faceted navigation* have the drawback of not being executable over unstructured text data. Additionally, the amount of documents has exponentially increased with the evolution of large domain-specific electronic repositories (e.g., arxiv.org [19]) as well as with the tremendous growth of the WWW. Besides, users rarely utilize boolean operators. They prefer to issue keywords as their initial access point [13,25]. Therefore, researchers developed more advanced approaches to search fulltexts [47]. Retrieval systems that enable fulltext search are usually classified under the term „information retrieval engine“. The term refers to any system that can identify relevant documents from a large corpus of texts whenever a user poses a keyword query that represents his/her information need [47]. A similar definition was proposed by Manning et al., who in their textbook on the topic characterized IR engines as follows [158].

„Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).“

IR engines primarily process documents and therefore need to provide natural language processing (NLP) units that identify relevant terms from unstructured text data. An index is set up with these terms, based on which user queries are answered.

One of the first steps of a common NLP workflow is the deconstruction of the char sequences of a document into separate units. This process is called *tokenization* and serves the purpose of omitting irrelevant chars, such as punctuation, as well as identifying semantically meaningful groups of chars. In the next stage, *stopword removal* is performed. It encompasses the exclusion of unnecessary words (e.g., „and“, „or“, „the“) which would not be of help during retrieval. Additionally, the engine eliminates superficial term differences which would otherwise prevent term-document matches. This process of token canonization is called *normalization*. Other normalization operations concern the handling of inflectional or derivational forms of a word. These differences can either be handled by *stemming* or by *lemmatization*. When *stemming* is applied, the system simply chops off a certain number of characters at the end of the word. *Lemmatization*, on the other hand, is a morphological analysis based upon which the base word form is returned for document indexing [158].

Upon execution of the beforementioned preprocessing steps, inverted indices are set up. An inverted index is the most frequently applied data structure in IR engines. It matches keyword queries with documents. The index lists document IDs for each item [47]. Upon matching keyword terms with index entries, the system determines how similar this match is. Thus, the system can rank search results according to relevance instead of showing an unordered set of items. IR engines often utilize a vector space model in combination with a TF-IDF weighting scheme (see Subsect. 2.2.2). This model represents each text document as a vector of words or phrases based upon which the system calculates the similarity between a document and a user query. TF-IDF weighting generates fairly good results [239].

Similarity-based document ranking is applicable to situations where users pose ad-hoc queries to a retrieval system. However, for some domains, such as digital library (DL) search, users might have standing information needs for which the system should be able to provide alerts or updates that indicate which of the newly added documents fall into their areas of interest. This kind of retrieval task can be modeled as a classification problem. Typical techniques in this category are naïve Bayes, support vector machines and other machine learning classifiers, such as decision trees, logistic regression approaches or neural networks [158].

However, these classification approaches are not particularly well-suited to tackle the task of *on-the-fly retrieval* from LOD repositories since they require a classification model before being able to answer user requests. TF-IDF ranking, on the other hand, does not depend on model building. Additionally, the application of IRIs allows relinquishing some of the previously mentioned NLP tasks of text-based preprocessing since IRIs uniquely identify resources.

Another technique from IR research is *faceted search*. Up until the 1990s, IR tasks could be either carried out with exact matching of user filters and document features or by calculating the similarity between user queries and document content. However, while the former approach does not provide any information on the relevance of the results, the latter lacks the possibility to filter items with regard to user preferences. Hence, *faceted search systems* were developed. They provide the missing link between the dichotomic approaches of *faceted navigation* and *similarity-based retrieval*, as they combine these techniques in a single user interface. When utilizing a *faceted search system*, users typically issue a keyword query. The IR engine matches these terms with the documents contained in the collection by accessing the inverted index of the system. Upon having identified matching items, similarity values between the documents and the user query are calculated for the final ranking. Additionally, for certain predefined and typical features (i.e., facets) of the document collection, the system displays how many of the matching results fall into each facet category [239]. This allows users to inspect a breakdown of their search hits with regard to certain features of the retrieved items. For instance, when searching for job postings, users can pose a query that attempts to capture the job title of the advertisements (i.e., „software engineer“). Afterwards, they are shown a ranked list of matching search results with additional filter options (e.g., city, industry, company name) and the total number of times this value occurs among the set of selected items [93].

Faceted search represents a powerful retrieval technique since it allows formulation of semi-structured queries that can capture a user’s inherent information need from different angles [239]. However, certain performance issues can occur when the engine processes multidimensional data. While the generation of single-valued facets is rather straightforward and can be facilitated by inverted index structures, the provision of multidimensional facets requires advanced indexing. For instance, to display pivot facets (e.g., the number of postings in a certain industry among the set of matching job postings at a specified location), the search engine library Apache Solr/Lucene needs to precache facet counts. Often, the precache takes up too much RAM and can potentially lead to performance losses. Another shortcoming of *faceted systems* is that they do not facilitate the search of multiple entity types, as they apply a limited data model of documents and facets. For instance, if the company type constitutes a facet, the job search engine can only query the corresponding features (e.g., no. of employees, sales range), when they have been assigned to the documents (i.e., job postings) as well. Hence, diverse query types need to be prepared by comprehensive indices and cannot be answered right away [93].

An additional drawback of *faceted search systems* is the limited expressiveness of filter conditions. While nested facets [93] as well as boolean facet combinations [239] are possible, more advanced query patterns, such as graph-based filters are not supported, due to the underlying table-like data structure. However, despite these limitations, *faceted search systems* offer helpful strategies for resource access and assist users in finding relevant items. In fact, scientists did not only prove the usefulness of *faceted systems* [25, 141, 239], but their high prevalence among major commercial applications, such as Amazon or eBay is also impressive evidence for the effectiveness of this retrieval approach [239]. The method could be extended to graph-based data and LOD-enabled RS as well.

Table 5.1 summarizes the previously mentioned aspects with regard to the system features of the requirements specification of Chapter 4. *RQ5-A* denotes an engine’s ability to apply simple filters, while *RQ5-B* stands for expressive query constraints. Apart from the possibility to formulate filter conditions for similarity-based results, there are no IR approaches that can be seamlessly transferred to the software engineering process of this thesis.

Table 5.1: Summary of IR approaches

Approach	Similarity Calculation			Query Facilities				Virtual Data Integr.	
	RQ1	RQ2	RQ3	RQ4	RQ5-A	RQ5-B	RQ6	RQ7	RQ8
Faceted Navigation / Parametric Search	X	X	X	X	✓	(✓)	X	X	X
Ranked Retrieval	X	X	X	X	✓	(✓)	X	X	X
Faceted Search	X	X	X	X	✓	(✓)	X	X	X

5.1.2 Query-based Recommender Systems

While IR systems are an interesting research direction to be investigated for LOD-enabled RS, other valuable approaches can be found in query-based recommendation. Researchers have

long been concerned with the limitations of CB, CF and KB systems in terms of customized recommendation retrieval [140]. Common approaches usually follow a user-item model and do not take other contextual aspects, such as the time or a user's companionship, into account [4]. For instance, a travel RS would probably have to suggest different destinations to the user in summer and winter [6]. While KB systems might be able to address this requirement, they lack the feature of adaptability to different domains. The LOD cloud, on the other hand, offers rich information sources, which could be applied as ready-to-use knowledge bases.

The requirements analysis stated that the LOD-enabled RS should provide a SPARQL-like query language that can formulate expressive recommendation requests. In the field of non-LOD RS, the idea of query-based recommendation retrieval has been around for some time now. Real-world RS usually operate on databases whose information could be of value for recommendations tasks. Koutrika et al. report that numerous students and administrators of their course RS, which runs on an e-learning database at Stanford University, requested customizable recommendations [140]. Other researchers have pointed out that query-based suggestions might not only be helpful for end users, but also for system providers [6].

Consequently, users (e.g., customers or administrators) should be enabled to express requests in the same way as they would query a conventional relational database. Hence, the recommendation model is not „hard-wired“ into the system, but users can retrieve suggestions using ad-hoc queries that resemble Structured Query Language (SQL) requests [4].

To this date, there only exists a small number of query-based RS. The first of this kind was the multidimensional RS, which was proposed by Adomavicius et al. in 2001. The central idea of the approach is to retrieve recommendations from multiple dimensions and aggregation hierarchies. The corresponding data schema resembles that of schemas commonly used in Online Analytical Processing (OLAP) cubes. However, the processor of a query-based RS has to be supplemented with an additional recommendation engine. The system of Adomavicius et al. can navigate along product hierarchies or can group customers thus basing recommendations on different granularity levels of typical RS parameters (i.e., users or items) as well as additional contextual factors (e.g., time or place) [4].

In addition to their application of the OLAP model for RS, Adomavicius et al. were also the first to introduce a recommendation query language (RQL). The RQL engine operates on top of an OLAP cube and retrieves recommendations through the execution of subsequent queries on database tables [4]. Ten years later, scientists from the same research group refined the approach by transforming RQL into the full-fledged query language REQUEST with a Backus-Naur form (BNF) syntax specification and additional algebra semantics. In their 2011 paper, the researchers elaborate on their reasons for developing an entirely new query language instead of relying on a non-integrated approach where SQL queries are combined merely with similarity calculation. They reasonably point out that SQL is a general-purpose language, with which users cannot express recommendation requests. Instead, they would have to formulate multiple SQL queries to identify items that might be of interest to them. In addition to producing cum-

bersome SQL queries, this requires additional cognitive effort on the side of the user [6]. As RQL, REQUEST is based on the OLAP paradigm and relies on dimensions (e.g., users, items, time or place), attribute hierarchies (e.g., product categories) and quantitative measures (e.g., ratings or implicit feedback information) to generate automated suggestions. Listing 5.1 shows an example REQUEST query from the paper by Adomavicius et al. [6]. It specifies a user's interest for movies that fit personal preferences and that are in line with certain conditions, such as the genre of the movie (Movie.Genre = „Action“) or the composition of the community of like-minded peers (User.Age >= 18).

Listing 5.1: Example REQUEST query

```
RECOMMEND Movie TO User
Using MovieRecommender
RESTRICT Movie.Genre = 'Action' AND User.Age >= 18
BASED ON PersonalRating(AVG)
SHOW TOP 5 BY PersonalRating
```

Koutrika et al. proposed another approach to integrated recommendation retrieval over relational databases. They introduced the FlexRecs system, which provides automated suggestions for courses in a university e-learning system. Koutrika et al. do not rely on an entirely new query language. Instead, they complement the SQL syntax with an „extend“ operator that creates a virtual nested relation. By these means, it is possible to adapt the search to specific requirements and to combine different RS approaches (e.g., CB and CF). Their approach decouples the actual retrieval process from the meta-level workflow specification that lies at the foundation of a complex recommendation request. FlexRecs can flexibly combine results from different recommendation algorithms, and users can define sequences of operations, with which items and users can be filtered both before and after the actual retrieval process [140]. While the query languages of Adomavicius et al. [4, 6] simply enable preference-based filtering with certain user specifications, Koutrika et al. introduced a new type of request that merges user constraints and common RS operations at different stages of the recommendation workflow. This is in line with Requirement 6 of the specification from Chapter 4.

However, the approaches of Koutrika et al. and Adomavicius et al. have the downside that they depend on rating data or comprehensive metadata descriptions, which makes empty result sets not unlikely due to *data sparsity* issues. Data is even sparser when the recommendation engine has to operate on a reduced set of items or users from a local database. Although Adomavicius et al. have pointed out that there exist possible workarounds for this shortcoming (e.g., application of matrix factorization to fill cells with missing ratings) this problem might still exist for highly selective recommendation requests and can produce low-quality results [6].

An additional point concerns the integration of data sources. Each of the presented approaches to query-based retrieval relies on the assumption that metadata, as well as user preferences, reside in the same repository. Nevertheless, this might not always be the case and would, therefore, require data migration.

Another shortcoming of the presented approaches is that they are limited to relational data and SQL-like queries. The strengths of LOD and expressive SPARQL queries (Sect. 3.2), on the other hand, might be even better suited for flexible query-based retrieval. Table 5.4 summarizes the key features of the approaches. In addition to the functional limitations, none of the presented query-based RS has been evaluated in a user study. Even though it seems reasonable to hypothesize that expressive queries improve recommendation results, it still has to be empirically verified.

Table 5.2: Summary of query-based RS

Paper	Domain	Similarity Calculation			Query Facilities				Virtual Data Integr.	
		RQ1	RQ2	RQ3	RQ4	RQ5-A	RQ5-B	RQ6	RQ7	RQ8
[4], [6]	General Purpose	X	X	X	X	✓	X	X	X	X
[140]	University Courses	X	X	X	X	✓	X	✓	X	X

5.2 Linked Data-enabled Systems

5.2.1 Linked Data Search Systems

There exist also some retrieval approaches that have been developed for Linked Data. Upon the publication of the first openly accessible datasets, researchers argued that indexing these resources would help users to find suitable vocabularies. Swoogle was among the first IR systems for the Semantic Web. The engine crawls and indexes ontological RDF data thus enabling the retrieval of vocabularies by keyword queries. It also provides simple filter options for query refinement [72].

Subsequent LOD search engines were designed to search instance data. However, during the mid-2000s, when the LOD cloud was still in an early state, RDF resources were rare. That is why Huynh et al. proposed a browser extension called PiggyBank which can be used to turn web documents into RDF data. The documents are stored in an Apache Solr/Lucene index and can be accessed by a common faceted browsing interface [114]. Both Swoogle and PiggyBank provide a similarity detection feature for Semantic Web resources. Other Linked Data search engines, such as Tabulator, Humboldt, mSpace, the facet and gFacet browsers or the VisiNav system are based on visual interfaces that assist users in exploring RDF graphs [33, 103, 105, 109, 135, 254]. Another example in this category is Aemoo. It extracts and displays RDF data using encyclopedic knowledge patterns (EKP). EKPs are fixed sets of properties [171]. The browsers help users navigate RDF graphs. Hence, even though these systems neither provide advanced search engine technology nor SPARQL syntax elements, they can assist in the formulation of expressive filter requests.

Aside from visual data browsers, researchers developed native Linked Data search engines with facet interfaces [53, 58, 96, 115, 246]. Some researchers have advanced the idea of faceted search in LOD repositories by accounting for the fact that semantic data allows for advanced

retrieval options. Oren et al. propose applying graph-based facet patterns (e.g., join operations or conjunctions) instead of simple item filters. They evaluated the feasibility of the approach in a small user study with 15 participants. It showed promising results in favor of graph filters [178]. Arenas et al. took the notion of expressive facets one step further: They developed OWL 2 reasoning options for RDF annotations. The approaches of Arenas et al. and Oren et al. are interesting because they make use of the strengths of ontological data (i.e., expressivity, and deductive reasoning capabilities) [17]. However, they fall short regarding ad-hoc querying. Other engines tackle LOD search with large-scale technologies. These research prototypes rely on parallel processing frameworks, such as Hadoop [177] and mimic cluster infrastructures of document-based web search engines (e.g., Google) to process RDF data efficiently [110]. Additionally, the engines provide strategies for large-scale reasoning tasks and are thus able to infer and index additional triple statements [110, 177]. However, these systems do not provide special data access strategies, since their focus lies more on backend design.

Other Linked Data search solutions were developed to facilitate *cross-repository retrieval* [97, 219]. For instance, the scientific retrieval engine by Sean et al. can generate query hits from different LOD repositories. The engine computes results in an ad-hoc fashion by matching the user's keywords with publication annotations from distributed collections. For this purpose, they apply cross-concordance links between different knowledge organization systems.

There are also systems that tackle the task of LOD search through spreading activation algorithms. Hence, depending on the similarity between the entities contained in a user request and their counterparts in the repository, RDF graphs are traversed by subsequently activating nodes and subgraph structures, which are likely to be relevant [81, 86, 128, 142, 162, 233]. Among these systems, the most interesting approach is Discovery Hub. This query engine accesses the DBpedia dataset through a random walk technique thus facilitating ad-hoc queries [162]. However, the downside of spreading activation engines is the randomness of search results [86, 142, 233]. The graph traversal engine may only be able to detect similar items if the triple processor happens to follow a fitting path. The Discovery Hub system considers this aspect to only some extent. While the interface allows users to define, which item features are most important to them, it does neither provide the possibility to specify the desired item types nor to formulate expressive filter conditions [73].

To have more control over the search process, system providers could follow the approach by Kaminskas and Fernández who have proposed executing search tasks over manually crafted RDF subgraphs. Their prototype provides *cross-domain* search results from DBpedia. However, with this approach, system providers would have to extract suitable RDF subgraphs from LOD repositories beforehand, which prevents ad-hoc recommendations.

In summary, it can be stated that existing Linked Data search engines already offer highly useful features. However, while visual data browsers lack the sophistication of native search systems and conventional index-based engines are not able to perform *on-the-fly* similarity calculation, spreading activation approaches might sometimes produce random results, due to insufficient

filter options and the unpredictability of their path traversal algorithms. Table 5.3 gives an overview of the findings from the literature survey on LOD search engines.

Table 5.3: Summary of Linked Data search engines

Paper	Domain	Similarity Calculation			Query Facilities				Virtual Data Integr.	
		RQ1	RQ2	RQ3	RQ4	RQ5-A	RQ5-B	RQ6	RQ7	RQ8
[10]	Music	X	X	X	X	X	X	X	X	X
[114]	Web Documents	X	✓	X	X	✓	X	X	X	X
[72]	Ontologies	X	✓	X	X	✓	X	X	X	X
[33]	General Purpose	X	X	X	✓	✓	(✓)	X	X	X
[135]	General Purpose	X	X	X	X	✓	(✓)	X	X	X
[254]	Music	X	X	X	X	✓	(✓)	X	X	X
[109]	Arts	X	X	X	X	✓	(✓)	X	X	X
[105]	General Purpose	X	X	X	X	✓	(✓)	X	X	X
[103]	General Purpose	X	X	X	X	✓	(✓)	X	X	X
[171]	General Purpose	X	X	X	X	✓	(✓)	X	X	X
[58]	General Purpose	X	✓	X	X	✓	X	X	X	X
[96]	General Purpose	X	✓	X	X	✓	X	X	X	X
[53]	General Purpose	X	✓	X	X	✓	X	X	X	X
[115]	General Purpose	X	✓	X	X	✓	X	X	X	X
[246]	Videos	X	✓	X	X	✓	X	X	X	X
[178]	General Purpose	X	✓	X	X	✓	✓	X	X	X
[177]	General Purpose	X	X	X	X	X	X	X	X	X
[110]	General Purpose	X	X	X	X	X	X	X	X	X
[219]	Scientific Publications	X	✓	(✓)	✓	X	X	X	✓	✓
[86]	General Purpose	(✓)	✓	X	X	X	X	X	X	✓
[142]	General Purpose	(✓)	✓	X	X	X	X	X	X	✓
[233]	Keywords	(✓)	✓	✓	X	X	X	X	X	X
[128], [81]	POI, Music	X	X	✓	X	X	X	X	X	X
[162]	General Purpose	(✓)	✓	X	✓	✓	X	X	X	✓

5.2.2 Linked Data Recommender Systems

Another research direction related to LOD-enabled retrieval are Linked Data Recommender Systems. Scientists developed the first LDRS with a different purpose than LOD search engines. While research on LOD search focuses on finding a better way for users to explore RDF resources, the initial LDRS utilized LOD resources to enhance the metadata of conventional content-based RS [66, 67, 168]. These systems match items from the user profile with their counterparts in the LOD cloud. Providers of music, movie or book RS can find a vast amount of metadata in DBpedia when mapping items with the corresponding IRI [52, 67, 186]. This process can be facilitated by matching discriminative features (e.g., a movie's title) with the related properties (`rdfs:label`) and literal values in the DBpedia [67]. By these means, certain widely used datasets from the RS community, such as the Lastfm and the MovieLens datasets, have been matched to DBpedia IRIs [57, 67, 179].

Researchers, who enhanced feature data with LOD, claim that they were able to boost the recommendation quality of their systems [66, 67, 127, 168]. Thus, common problems of CB RS, such as *limited content analysis*, could be tackled with additional resources from the web of data. Most LDRS determine similarity values for each item pair offline by applying natural language processing (NLP) techniques. Some researchers propose utilizing topic modeling approaches, such as Latent Dirichlet Allocation (LDA) [127] or Latent Semantic Indexing

(LSI) [168], on RDF data to detect the underlying concepts, which characterize the respective items. However, one of the strengths of the LOD cloud is that item features are often uniquely identified by IRIs. Hence, common problems of traditional text-based IR systems, such as synonymous words or polysemy do not necessarily have to be handled by the recommendation engine [66, 152].

For this reason, other researchers propose relinquishing the laborious process of topic identification and to rely on the term-document matrix of certain item features instead. In this context, a common strategy is to count the number of matching features among the set of all features of two items [106, 112, 234, 259] and apply a TF-IDF weighting scheme on feature data. With this approach, the similarity value is high when two items share many features which rarely occur in the repository [66, 67, 97, 129, 208]. The TF-IDF scheme takes into account the number of times a term is mentioned in the metadata descriptions (i.e., the TF-value). However, many LOD resources are annotated with IRIs, which mostly occur once in a single feature set of an item. Hence, in many cases, TF-values can be omitted without influencing the outcome. In concordance with that, Meymandpour et al. propose simply computing item-to-item similarity values through the summation of the information content (IC) (i.e., a metric, which closely resembles IDF) of matching item features [165].

Items are usually characterized by different feature types (e.g., the genre vs. the record label of a music album) which might not be equally relevant for predicting user interests. Therefore, researchers developed approaches to determine the predictive strength of properties. For instance, Peska et al. maximize a linear regression model that predicts item similarity to assign appropriate weights to each property [186]. Other LDRS apply meta-learning and meta-heuristic approaches, such as genetic algorithms, particle swarm optimization or stacking to determine the best weight configuration [67, 129, 206, 249].

Aside from NLP methods, item-to-item similarities can also be detected by graph-based metrics. The *Linked Data Semantic Distance* (LDS) was one of the first measures of this kind. It tracks the number of direct/indirect and incoming/outgoing links between two RDF resources [182]. However, the application of the metric might be too expensive for *on-the-fly retrieval*. In performance experiments, the generation of recommendations often took up dozens of seconds with LDS, even though the similarity detection engine operated on a reduced DBpedia dataset [182]. Hence, more advanced graph-based metrics, e.g., by Grouès et al. [94], Maedche et al. [156] or Harispe et al. [99] may be even less scalable.

Apart from NLP- and graph-based approaches to similarity calculation, there exists another type of LOD-enabled RS, which predominantly relies on pure SPARQL-based enhancement. For instance, the RS by Ahn et al. and Ozdikis et al. utilize LOD resources to complement metadata information of result lists that are presented to the user to help him/her make an informed decision about the relevance of the recommended items [10, 180]. Mannens et al. apply LOD enrichment after similarity calculation. They use RDF resources to facilitate simple faceted filtering of recommendation lists. The travel RS by Varga et al. retrieves geo-related data snippets

5.2.3 Query-based Linked Data Recommender and Search Systems

In addition to IR engines for LOD and LOD-enabled RS, there exist some systems that utilize SPARQL in combination with similarity calculation for retrieval tasks. An early and interesting example for such a system is the Corese Search Engine. It consists of an RDF query language and an IR engine. The system can retrieve items, even if they do not exactly match the specified user requirements [55]. Another query-based LDRS is the iSPARQL engine by Kiefer et al. It was initially developed to facilitate data integration tasks, but can also be applied for preference-based search. Kiefer et al. extend the SPARQL syntax with a pattern that defines operations to identify similar items [130].

However, with both Corese and iSPARQL, users cannot specify their preferences. Therefore, other researchers focus on the integration of more nuanced profile information into the SPARQL-based retrieval process. Siberski et al. propose extending the SPARQL syntax with an additional solution modifier with which end users can personalize the search [227]. Guerousova et al. demonstrate that qualitative preference information can be integrated into native SPARQL queries without modifying the SPARQL 1.1 syntax by merely using OPTIONAL graph patterns [95]. Nevertheless, each of these approaches relies on exact matchings of preference specifications. Hence, user conditions can quickly become too restrictive, which is a significant shortcoming, as profile data is still only an approximation of what the user might want to retrieve [207].

Therefore, Rosati et al. propose to group preference data into hard or soft constraints. While hard constraints strictly filter result sets, soft constraints contribute to the ranking of items, instead of excluding them. In this context, the researchers introduce a lightweight ontology, with which users can express their preferences concerning different attribute-value configurations. From these preferences, a *ceteris paribus* (CP) network is constructed and matched with the triple statements in the repository. The method ranks resources and favors items that are most in line with the preference statements of the user. The query system by Rosati et al. represents a highly advanced approach in the area of preference-based LOD retrieval, but also has some limitations. Since the system works with CP-networks, lengthy processing times have to be expected [43]. Another aspect concerns the preference elicitation process. It requires both a lot of effort and domain proficiency on the side of the user. As has been argued before, knowledge-based RS perform best in rarely-occurring decision situations with highly specialized items. The LOD cloud, on the other hand, provides metadata for many domains (e.g., multimedia retrieval) where a straightforward CF or CB recommendation strategy often fully suffices.

In another line of research on query-based LDRS, scientists propose storing user preference data in conjunction with triple statements from RDF repositories [23, 189]. These systems do not provide options for ad-hoc access of LOD datasets since they require integration of preference data and RDF-based information before executing the recommendation tasks. An example

for this category of RS engines is the system by Policarpio et al. It can generate CB- or CF-based suggestions from RDF data, but it still lacks an integrated SPARQL-like query language which can pose recommendation requests in a single query statement [189]. Ayala et al. address this research gap with their RecSPARQL system. It provides a query language that relies on SPARQL elements and RS operators to generate suggestions. To this date, the RecSPARQL system is the only one that transfers the idea of query-based recommendation retrieval from the context of relational databases to RDF repositories [23]. However, it does not yet take full advantage of the expressiveness of the RDF data model and the potential to formulate advanced retrieval requests. For instance, the notion of workflows [140], which enables execution of similarity calculation at different stages of the retrieval process is not realized in this approach. Additionally, from the examples the researchers give in their RecSPARQL paper, it can be seen that the data is stored in a table-like fashion [23]. On top of that, the distributed nature of the LOD cloud demands an API-based solution to leverage the richness of the data web. In this context, the integration of user preference with item feature information before recommendation retrieval is a cumbersome additional processing step required by the RecSPARQL system. Table 5.5 shows the key findings of the literature survey on query-based LDRS.

Table 5.5: Summary of query-based LDRS and search systems

Paper	Domain	Similarity Calculation			Query Facilities				Virtual Data Integr.	
		RQ1	RQ2	RQ3	RQ4	RQ5-A	RQ5-B	RQ6	RQ7	RQ8
[55]	General Purpose	✓	✓	✓	✗	✓	✓	✗	✗	✗
[130]	Data Integration	(✓)	✓	✓	✓	✓	✓	✗	✗	✗
[227]	General Purpose	✗	✗	✗	✓	✓	✓	✗	✗	✗
[95]	General Purpose	✗	✗	✗	✓	✓	✓	✗	✗	✗
[207]	General Purpose	✗	✗	✗	✓	✓	✓	✗	✗	✗
[189]	General Purpose	✗	✓	✗	(✓)	✓	✗	✗	✗	✗
[23]	General Purpose	✗	✓	✓	✓	✓	✗	✗	✗	✗

5.3 Summary & Research Agenda

The survey on RS and retrieval systems has shown that there are already methods for personalized search in place that are relevant for the requirements of this thesis. In the category of non-Linked Data systems, there exist text-based search engines that perform relevance rankings for unstructured resources and offer means to state filter conditions to complement search queries. On the other hand, some systems can issue personalized SQL queries over relational databases to provide recommendations. Both retrieval strategies can execute customized user requests. The strength of search engines is their ability to quickly perform query matchings and similarity-based rankings due to extensive preprocessing and indexing of resources before runtime execution. However, low latency of recommendation results comes at the cost of a „hard-wired“ data model, which prevents individual customizations and causes limitations concerning up-to-date retrieval.

The strengths of query-based RS, on the other hand, are their expressive filtering options. How-

ever, these approaches have the drawback of data sparsity since the data space of users and items can be tremendously reduced when only extracting information from a local database.

Despite the strengths and weaknesses of the discussed approaches, it also has to be taken into account that they are not designed to consume LOD. Many Linked Data-enabled retrieval engines, on the other hand, already leverage some of the hidden potentials of openly available RDF resources to generate automated suggestions. In the category of LOD search engines, the systems that enable expressive graph-based facet filters are promising examples. Another interesting research direction involves techniques of *on-the-fly retrieval* that apply a spreading activation algorithm to process the RDF graph. Despite these approaches, conventional LOD search engines are not designed to consume profile data (i.e., explicit/implicit preferences for specific items). Hence, it is not surprising that they do not adequately address the system requirements that were specified in Chapter 4. However, most of the existing LOD-enabled RS do not tackle many of the specified requirements either. Besides enhancing user profile information with RDF data from open repositories, these systems often do not address other strengths of the LOD technology stack, such as *expressive query* options or remote *cross-repository retrieval*. In the category of query-based LDRS, only a few systems take advantage of the query capabilities of SPARQL as well as the richness of RDF repositories. Nevertheless, these systems fall short regarding scalable *on-the-fly similarity calculation* and profile-based retrieval from SPARQL endpoints. Both shortcomings can be attributed to the fact that in most query-oriented LDRS, user preferences and metadata information reside in the same repository. However, this goes against the notion of the LOD cloud as a decentralized and openly accessible data space which can be queried through API-like interfaces.

In summary, existing retrieval and RS engines already provide useful features for personalized resource access. Nevertheless, in terms of using the LOD infrastructure to leverage recommendation effectiveness, there does not yet exist an approach which provides all of the required features in a single solution. Additionally, some of the previously specified requirements have only been researched to a limited extent (e.g., *on-the-fly similarity calculation*), while the demand to facilitate LOD-enabled *advanced recommendation requests* has not been tackled at all. The following chapter will introduce a recommendation engine which draws upon some of the presented approaches but also seeks to address the open research question of efficient and effective LOD retrieval through the fulfilment of the requirements of Chapter 4.

6 The SKOS Recommender

This chapter introduces the SKOS Recommender that implements the requirements specification of Chapter 4. It presents the foundation of the system (Sect. 6.1) as well as the usage scenarios and LOD repositories with which it will be tested during the development process (Sect. 6.2). Sections 6.3-6.8 explain the functioning of the system features and provide results for example recommendation requests. Section 6.9 gives details on the implementation of the engine.

6.1 Simple Knowledge Organization Systems

As has been outlined in the previous section, many LDRS apply content-based recommendation methods on item feature data. Similar items are identified by processing triple statements from an RDF dataset. Items are mostly represented by IRIs and item features are either an element of the set of literals (L) or the set of IRIs (I) [66, 67, 129, 168, 179, 186, 206, 259]. Many authors have contrasted traditional recommendation strategies with LDRS approaches [182, 186, 259] to validate the suitability of LOD for these tasks. Others have also attempted to compare several Linked Data-enabled approaches against each other. Some tests showed that subject annotations (e.g., as indicated by the property `dct:subject` or `dc:subject`) are often the best feature type for content-based recommendations [66, 67]. This finding might be rooted in the long-standing tradition of subject indexing to describe an item in such a way, that the user will most easily find it [113].

Usually, LOD resources are annotated with numerous different feature types. Consider the example in Figure 6.1 for illustration. It shows the DBpedia resource `dbr:Lord_of_the_Flies`. The resource represents a real-world book item. Different kinds of terms (literals or IRIs) and base types (i.e., date and string) serve as metadata features for this LOD resource.

This diversity requires identification of suitable features because not every feature type might be relevant for similarity calculation. However, the selection process can be error-prone and time-consuming [249]. Additionally, *on-the-fly retrieval* is hindered when item features have to be manually selected. Thus, reliance on subject annotations is a viable approach to generate recommendations from heterogeneous data sources in an ad-hoc fashion (RQ1 and RQ2). Often, datasets declare subject annotations with the help of the DCMI terms [62]. Besides vocabularies such as RDF and OWL, the DCMI vocabulary has become a de-facto standard for describing resources thereby adhering to LOD best practices. DCMI terms occur in more than half of

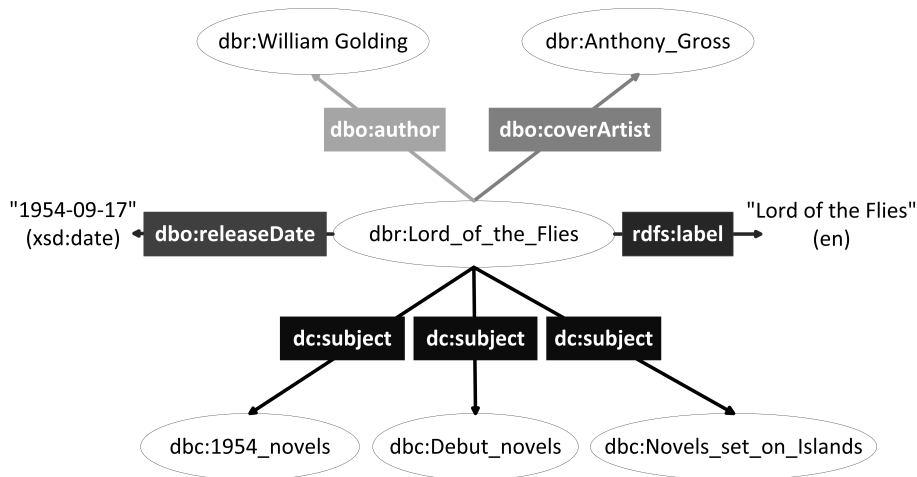


Fig. 6.1: Item features of an example LOD resource

the datasets in the LOD cloud [215]. The definition states that data publishers should use the properties `dct:subject` and `dc:subject`¹ for resource description as follows [62]:

„Typically, the subject will be represented using keywords, key phrases, or classification codes. Recommended best practice is to use a controlled vocabulary.“

In libraries and other information-providing institutions, controlled vocabularies have long been used to organize and facilitate access to large document collections. Different subcategories of these vocabularies or knowledge organization systems (KOS) have evolved and are known as thesauri, classification schemes, taxonomies, subject heading systems or taxonomies. IR systems apply them to search for both printed and electronic document collections [166]. Although differing slightly in nature, these subcategories share important features and engines utilize them in similar use cases [117]. For instance, all knowledge organization systems distinguish between concepts and terms. A concept represents an idea that is unambiguously identified and characterized by terms [158]. Thus, annotations conforming to the DCMI recommendation to use KOS concepts enable efficient similarity calculation [67]. In contrast to that, natural text annotations require time-consuming preprocessing operations [168, 206], which prevent ad-hoc retrieval. Hence, it is proposed to take advantage of KOS annotations to meet the functional requirement of *on-the-fly recommendations* (RQ1).

In the LOD cloud, the Simple Knowledge Organization System (SKOS) vocabulary is a de-facto standard for expression of controlled vocabularies in machine-readable form. SKOS systems are RDF-based terminologies that can be published and accessed on the web. SKOS concepts are uniquely identifiable IRI resources with each member being an instance of `OWL:Class`. They have a scope, a meaning, and concept label descriptions (i.e., `skos:prefLabel`, `skos:`

¹The namespaces PREFIX `dct:` <<http://purl.org/dc/terms/>> and PREFIX `dc:` <<http://purl.org/dc/elements/1.1/>> refer to sets of terms that can be used for metadata descriptions. The former namespace comprises all metadata terms maintained by DCMI, while the latter only contains a subset of this standard [62].

altLabel). Thus, vocabularies can distinguish homonymous as well as synonymous terms. Additionally, the SKOS standard provides expressions to relate concepts to each other, either in a hierarchical (i.e., with `skos:broader`, `skos:narrower` links) or in a non-hierarchical fashion (i.e., with `skos:related` links) [166]. This feature can be exploited for term expansion, as the similarity processing unit may incorporate topically related SKOS concepts. Thus, recommendation retrieval can be flexibly adapted to explore the RDF dataset more deeply with changing sets of subject descriptors (RQ3). SKOS vocabularies adhere to key principles of LOD publishing because they facilitate mappings of concepts across different SKOS systems (i.e., with `skos:exactMatch`, `skos:broadMatch` links), [38]. These cross-concordance links can leverage retrieval possibilities since they help to identify relevant resources across collections that apply different SKOS vocabularies [163] (RQ7). Figure 6.2 depicts the key components in the SKOS data model according to Baker et al. [24].

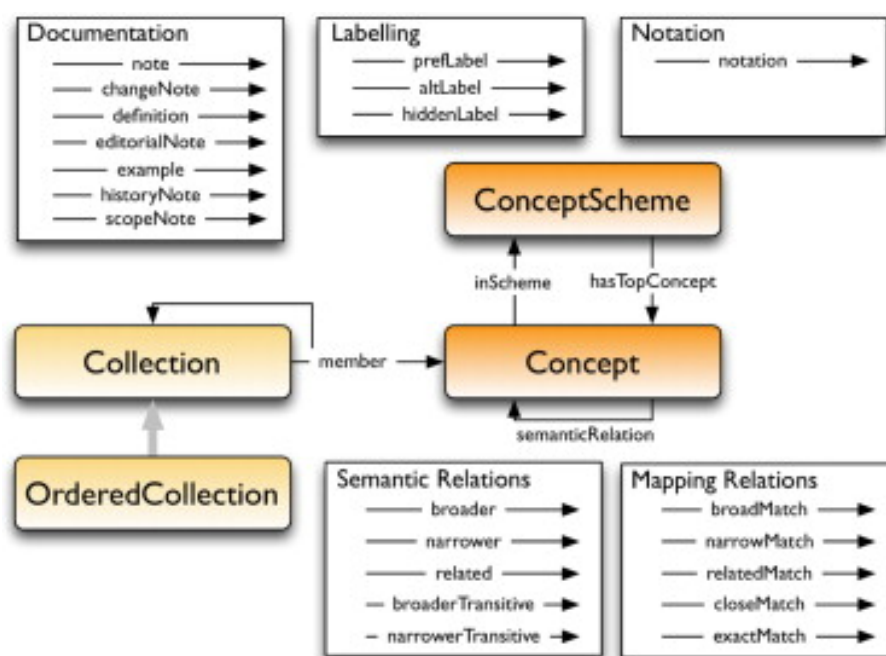


Fig. 6.2: Main elements of the SKOS data model [24]

During the last decade, data providers have released dozens of SKOS systems to the general public. Some of them facilitate bibliographic control for general-purpose collections (e.g., Library of Congress Subject Headings (LCSH)) [235], while others are domain-specific. Domain-specific vocabularies exist in various areas, such as economics (e.g., the Standard Thesaurus Economics (STW) [173]), social sciences [229] or life sciences (e.g., the AGROVOC Agricultural Thesaurus [48]).

The Linked Open Data crawl of 2014 found that the SKOS namespace occurred in more than 140 datasets (14%) of the entire LOD cloud, thereby being among the ten most frequently used vocabularies at that time [215]. In November 2016 the author issued a search query in

the Open Data repository Datahub and found that more than 2,400 datasets were tagged with „skos“ [220]. Additionally, DBpedia as one of the largest and most well-connected datasets in the LOD cloud applies concepts from a category graph in SKOS format to annotate its page resources [74]. Hence, it can be claimed that SKOS is a widely adopted standard in the LOD cloud which might be exploited to target challenges of data heterogeneity (RQ2). On top of that, through unique identification of subject descriptors, recommendations can be computed *on-the-fly* without requiring NLP operations.

6.2 Usage Scenarios

6.2.1 Scenario Selection

The to-be-developed technologies need to be applicable to numerous LOD collections and should provide helpful recommendations throughout different domains. Only then can it be assumed that the system offers an added value to users. Hence, conclusions concerning recommendation effectiveness are subject to evaluations within the context of several application scenarios. In this line of argumentation Aggarwal points out that [7]:

„Testing over multiple data sets is particularly important for assuring greater generalization power of the RS so that one can be assured that the algorithm works under a variety of settings.“

If subsequent evaluations will be carried out in multiple usage scenarios, it is better to consider the applicability of the methods in different domains throughout the development process as well. Hence, system features can be immediately tested for particular recommendation requests. Therefore, adequate usage scenarios have to be selected. Ricci et al. categorize RS into the following five types of recommender systems that were already presented in Section 2.1:

- Entertainment RS
- Content RS
- E-Commerce RS
- Service RS
- Social RS

Naturally, it would be desirable to have access to example datasets for these categories of RS in the LOD cloud. However, the data web does not contain representative collections for each RS type. For instance, there exist no high-quality RDF resources that would enable *social RS*

for networking sites. Firstly, most social LOD collections do not use the DCMI standard [215]. Another reason is that these datasets predominantly describe connections between agents (e.g., persons or organizations) [83]. Thus, they lack the required metadata descriptions for similarity calculation. Due to these shortcomings, social RS cannot be covered by an example usage scenario.

Fortunately, the LOD cloud provides high-quality data sources for *service RS*. For instance, DBpedia contains comprehensive metadata descriptions for places, which can be applied in the example usage scenario of travel destination search [204, 245].

Suitable use cases for *entertainment* and *e-commerce RS* are multimedia retrieval tasks. While in *entertainment RS* consumption of multimedia items (e.g., movies, music or books) is promoted, *e-commerce RS* are applied to increase product sales. For these cases, the LOD cloud holds rich metadata descriptions in the DBpedia repository. Hence, the domain of multimedia recommendations is a suitable usage scenario, since it both represents the RS landscape and is well covered by the web of data [245].

The usage scenario of digital library (DL) search can serve as an excellent example for *content RS*. While other potential application areas in this category would be news article or web page recommendations, there exist no suitable LOD resources for these domains. On the other hand, the LOD cloud holds many bibliographic datasets for DL retrieval [215], of which the search for economics publications can serve as an example. There are several reasons for this choice. Firstly, there exists a digital repository that provides access to openly available economics papers (i.e., EconStor) and corresponding metadata descriptions in the LOD cloud. Hence, the evaluation of the engine can be directly carried out for the real-world scenario of publication search. Another reason is that the EconStor dataset represents a well-maintained and up-to-date LOD repository, whose resources are annotated with concepts of the Standard Thesaurus Economics (STW) [41]. The STW is a SKOS thesaurus and openly available in the LOD cloud. It provides concept matchings that link from the STW Thesaurus to other SKOS vocabularies, such as the AGROVOC Thesaurus [173]. Indexing terms from the AGROVOC thesaurus are, in turn, used in other LOD repositories, such as the bibliographic database AGRIS [24, 48]. Hence, subsequent evaluations can test the system in the context of *cross-repository recommendations* (RQ7 and RQ8) as well.

In addition to being representative of typical RS tasks and LOD repositories, the literature survey on LDRS has shown that LOD enhancement in the specified domains is beneficial for content-based recommendations (Sect. 5.2). Each of the selected usage scenarios can accompany the development process and provide example queries that demonstrate the feasibility of the novel approaches to LOD-enabled retrieval. The multi-scenario approach avoids domain-specific biases and ensures that the engine's ability to offer useful suggestions is validated from different points of view. Hence, conclusions about system performance are based on a diversified set of requests. Thus, the author can explore the to-be-developed technologies from a broad range of information needs which are likely to differ among the selected usage scenar-

Table 6.1: Overview of the usage scenarios

Domain	Subdomain	LOD repository
Digital Library	Economics Agriculture	EconStor AGRIS
Travel		DBPedia
Multimedia	Movie Music Book	DBPedia DBPedia DBPedia

ios. Consequently, test queries will be adapted to the respective domains. The multi-scenario setup additionally requires the application of different datasets for recommendation retrieval to increase the generalizability of the results. Table 6.2 lists each domain and the corresponding LOD repository to be used in subsequent evaluations.

6.2.2 LOD Repositories

This subsection will describe the LOD repositories of the usage scenarios in detail. The DBpedia dataset is the most important among them. DBpedia resources are freely available on the data web. As advocated by the LOD community, its resources are uniquely identified by dereferenceable HTTP IRIs, which enables users to retrieve triples for entities through any browser. Users can also obtain DBpedia information from SPARQL endpoints [253].

However, dependence on a public API is not desirable for test runs. For instance, the DBpedia providers limit the number of queries that can be posed by a single IP address within 24 hours [197]. In addition to query limits, the public web server is sometimes unavailable throughout the day [245]. Hence, to ensure that system tests could be carried out uninterrupted during the development process, a local mirror was set up for the present study. The DBpedia community releases current versions of the dataset on a regular basis. At the time of writing of this thesis, the last data extraction took place in October 2016. It can be downloaded from the DBpedia site. The dataset contains more than 23 million triple statements from 125 language versions of Wikipedia. The English version alone describes 6.6 million entities [75]. It represents the most extensive version among all DBpedia language editions [60]. However, for test runs and experiments, a less recent release of the English DBpedia was applied, since the research project of this dissertation began in 2013 and initial tests were carried out on the latest English edition at that time (i.e., DBpedia 3.9). Hence, because of compatibility reasons, this version was utilized throughout the entire development process. It contains fewer data sources, than the most current version, but it is still very extensive. The author loaded the English DBpedia into a local OpenLink Virtuoso server [3]. Table 6.2 lists the number of available DBpedia entities with SKOS annotations in the respective domains.

²PREFIX swc: <http://data.semanticweb.org/ns/swc/ontology#>

³PREFIX purl: <http://purl.org/dc/elements/1.1/>

Table 6.2: Entity types in the usage scenarios

Domain	Subdomain	Entity Type (<code>rdf:type</code>)	# Entities
Digital Library	Economics	<code>swc:Paper</code> ²	81,598
	Agriculture	<code>purl:Article</code> ³	7,635,333
Travel		<code>dbo:Place</code>	725,546
Multimedia	Movie	<code>dbo:Film</code>	90,063
	Music	<code>schema:MusicGroup</code> ⁴	86,013
	Book	<code>dbo:Book</code>	31,172

The author identified entity types by `rdf:type` class assignments. In this context, it was ensured that classes were generic enough. For instance, the DBpedia Ontology provides classes for the three entity types movie (i.e., `dbo:Film`), book (i.e., `dbo:Book`) and place (i.e., `dbo:Place`), but not for music acts. Instead, it lists solo artists (`dbo:MusicalArtist`) and bands (`dbo:Band`) at different points in the class hierarchy [61]. Fortunately DBpedia entities are linked to classes from other ontologies as well. For instance, another type hierarchy provides the class `schema:MusicGroup`, which annotates both music groups and solo musicians. For this reason, this type was chosen to identify musical acts.

A multitude of attributes describe DBpedia entities. Depending on the type and the respective Wikipedia infobox template, attributes can range from geo-coordinates (e.g., for locations) to birth dates (e.g., for persons) or genres (e.g., for multimedia items) [76]. In addition to the different property types, DBpedia contains SKOS-based annotations for almost all its entities [169]. By default, the property declaration `dct:subject` is utilized for this purpose. Thus, for subject annotations the dataset adheres to the DCMI standard [76]. Subject categories originate from Wikipedia. Categories describe what an article is about and they help users to find related articles on similar topics [252]. DBpedia stores the hierarchical structure of the Wikipedia category system as a loose, polyhierarchic SKOS thesaurus without transitive dependencies [76, 166].

Both the article annotations and the SKOS-based structure of DBpedia can be utilized for similarity computation. However, there exist some quality issues in the DBpedia category graph (e.g., cyclic dependencies [155]) which need to be handled during runtime execution.

The DBpedia dataset is available under the Creative Commons Attribution-ShareAlike 3.0 License (CC BY-SA) and the GNU Free Documentation License [60]. Both licenses are typical „Open Data“ copyleft licenses, i.e., they allow users to share and modify data sources, but require that derived works are republished under the same terms [21, 91].

Aside from DBpedia, the LOD repository EconStor was applied during the development process. The repository provides bibliographic data for the DL usage scenario. It contains metadata descriptions of research papers and is the LOD edition of the corresponding fulltext Open Ac-

⁴PREFIX `schema:` <<http://schema.org>>

cess server. As of July 2017, the EconStor repository made more than 140.000 fulltexts publicly available [78]. It is one of the largest Open Access repositories in the domain of economics, as it enables self-archiving and ensures long-term access to pre- and postprint publications from the domain of economics. The Leibniz Information Centre for Economics (ZBW) in Kiel (Germany) provides the technical infrastructure and takes care of metadata maintenance tasks for both EconStor and the LOD edition [41]. EconStor LOD has a Public Domain license [79]. The license ensures that third parties can use and modify the data collection for their purposes without being bound to any copyright or copyleft regulations or restrictions regarding commercial use [195]. EconStor LOD was loaded into a local OpenLink Virtuoso server as well. The dataset is in beta status because the providers want to demonstrate that the release is still in development stage. At the time of writing of this thesis, the latest version of the EconStor LOD data dump was from May 2016. It contains metadata descriptions for thousands of papers [42]. EconStor LOD resources are annotated with the property `dc:subject` and link to STW descriptors [41]. The vocabulary covers the fields of (business) economics, industry sectors and related areas, such as politics and geography. The ZBW gradually transferred the thesaurus into a SKOS vocabulary in 2009 [173]. The SKOS system is published under the terms of the Open Database License (ODbL) [232]. The ODbL is a typical copyleft license that allows for free sharing as well as modifications, as long as derived versions are redistributed under the same terms [175]. As of July 2017, the thesaurus contained 6.000 descriptors and more than 20.000 synonymous expressions which can facilitate topic retrieval in economics databases. In contrast to the SKOS category graph of DBpedia, the STW defines transitive hierarchical relationships. Hence, the concept graph can be explored more extensively than its DBpedia counterpart (see Sect. 6.5). In addition to transitive relationships, the thesaurus also contains concordance links to other SKOS vocabularies on the LOD cloud.

Among them are links that connect STW concepts to AGROVOC descriptors [173]. AGROVOC is a comprehensive multilingual thesaurus. It was originally developed in the 1980s by the Food and Agriculture Organization of the United Nations (FAO) to enable indexing and retrieval of publications from the domains of agriculture, economics, and rural development.

With the advent of the Semantic Web, the FAO provided the AGROVOC thesaurus as a publicly accessible SKOS vocabulary [48]. AGROVOC is published under the Open Data Creative Commons 4.0 license [9]. It allows free redistribution and sharing of datasets even for commercial purposes, as long as the new version mentions both the original creator and potential changes [20].

In its current version, AGROVOC describes 33,038 entities in 33 languages. It is an important tool for knowledge organization in its field. Among others, the International System for Agricultural Science and Technology (AGRIS), which is a major bibliographic reference source for publications on agriculture, fisheries, forestry and environmental research, applies the AGROVOC thesaurus [24, 48, 51, 242]. The technical infrastructure of the FAO serves as a data hub for AGRIS, which receives data from more than 150 third party providers in over 65

Table 6.3: LOD repositories to be applied in the usage scenarios

LOD repository	Version	Annotat. property	SKOS Vocabulary	License
DBpedia	3.9 (2013)	dct:subject	SKOS Article Categories	CC BY-SA
EconStor	2016	dc:subject	STW	Public Domain, ODbL
AGRIS	2015	dct:subject	AGROVOC	CC BY

countries [51]. At the time of writing of this thesis, the latest public release of AGRIS registered metadata on more than 7.5 million articles (see Tab. 6.2). Since 2013, AGRIS data has been made publicly available in the LOD cloud. Hence, bibliographic information from AGRIS can be retrieved from the web of data including AGROVOC subject annotations [51]. These data sources can be exploited in conjunction with the SKOS mappings from the STW thesaurus to facilitate *cross-repository recommendation* retrieval. The latest edition of AGRIS is from 2015. It is available under the Creative Commons license 3.0 (CC BY 3.0) [8]. The AGRIS dump was loaded into an OpenLink Virtuoso server as well. Table 6.3 lists the features of the LOD repositories that were accessed throughout the development process and the evaluation. The following sections will introduce the technologies of a novel LOD-enabled RS, named SKOSRecommender (SKOSRec). The engine implements the functional features of the requirements specification from Chapter 4 by taking advantage of SKOS vocabularies and LOD repositories.

6.3 On-the-Fly Recommendations

One of the fundamental features of the SKOSRecommender is the ability to generate *on-the-fly recommendations* from LOD repositories. The approach is based on SKOS annotations. The engine identifies annotations by matching corresponding annotation properties in the RDF dataset (Definition 6).

Definition 6 (Annotation property). *An annotation property is an IRI, that is defined in the DCMI specification for subject annotations (Listing 6.1). It can occur in conjunction with one of the two namespaces, which are stated by the standard [62].*

Listing 6.1: Annotation property

```
<ANNOT.PROP> ::= dc:subject | dct:subject
```

These properties retrieve SKOS annotations for items preferred by a user (Definition 7).

Definition 7 (SKOS annotation triple). A SKOS annotation triple (st) is a triple that is comprised of an IRI in the subject position, an annotation property ($ANNOT.PROP$) as predicate and a concept c in the object position. The concept is part of a knowledge organization system S in SKOS format ($C = \{c \mid c \in S\}$) (Eq. 6.1).

$$st \in I \times \langle ANNOT.PROP \rangle \times C \quad (6.1)$$

SKOS annotation triples can quickly retrieve potential similar items. Therefore, the engine evaluates shared features between resources from the SKOS annotation dataset (Definition 8).

Definition 8 (SKOS annotation dataset). A SKOS annotation dataset (AD) is a subset of an RDF dataset that contains SKOS annotation triples (Eq. 6.2).

$$AD \subset I \times \langle ANNOT.PROP \rangle \times C \quad (6.2)$$

Before the similarity calculation process starts, the engine determines SKOS annotations for each item (r) from the user profile. The notion of SKOS annotations is specified in Definition 9.

Definition 9 (SKOS annotations). In the annotation graph, LOD resources directly link to concepts of a SKOS system via a predefined annotation property. SKOS annotations for an input resource r are defined as in Equation 6.3.

$$Annot(r) = \{c \in S \mid \exists (r, \langle ANNOT.PROP \rangle, c) \in AD\} \quad (6.3)$$

The corresponding SPARQL query that obtains SKOS annotations for a given LOD resource is depicted in Listing 6.2

Listing 6.2: Annotation query

```

1 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2 PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
3
4 SELECT DISTINCT ?c
5 WHERE
6 {
7   <r> <ANNOT.PROP> ?c .
8   ?c rdf:type skos:Concept .
9 }
```

Upon extraction of SKOS annotations for the input resource, the system determines which of the other resources in the dataset shares annotations with it. The engine identifies similar LOD

resources after having performed the before mentioned operations. In a database context, the system would calculate item-to-item similarity values in offline mode, since this procedure typically represents a performance bottleneck. However, in CB engines the feature matrix is usually sparse. Thus, similarities only have to be computed for items that share data points [67, 212]. This approach can be improved upon in LOD-enabled RS. Triples can be efficiently joined along item features, since native RDF storage systems usually index triples rather than columns [174]. Thus, the quick identification of mutual annotations through SPARQL requests facilitates ad-hoc similarity calculation. The query matches potentially relevant resources by finding items that share SKOS annotations with the input resource. By this method, the engine calculates similarity values on a reduced item set.

LOD resources are relevant when they share at least one feature (SKOS annotation) with the input resource. This approach follows the notion of standard metrics (such as *pearson correlation* or *cosine similarity*), which base similarity on matching features. However, in contrast to the mentioned measures, the SKOSRec engine does not consider all annotations of related resources, but only those that match the SKOS annotations of the input resource. The method reduces the number of triple statements to be processed and prevents high throughput rates. Hence, unlike conventional feature-based metrics, such as *cosine similarity*, similarity values are not normalized. Against the background of the requirement of *on-the-fly retrieval* (RQ1), the performance aspect is more critical than normalization. This assumption is in line with findings from other authors. For instance, Meymandpour and Davis propose a hybrid similarity metric for LOD. It computes the summated IC of the intersection of annotations of two LOD resources [165]. Mistry and Pavlidis conducted a comprehensive study on the performance of several metrics for computation of gene similarities. Their investigation revealed that a simple non-normalization measure named *term overlap measure* (TO) was competitive with the corresponding normalized metric (NTO) regarding *accuracy*, while being faster at the same time [167]. Hence, in the SKOSRec engine, non-matching SKOS annotations are omitted from similarity calculation in favor of quick processing times. Definition 10 specifies the notion of relevant resources and their annotations that are needed to identify similar items.

Definition 10 (Relevant resources and their annotations). *The conjunctive graph pattern P_r matches all items and subjects that are potentially relevant for recommendation retrieval (Eq. 6.4).*

$$P_r = (r, \langle \text{ANNOT.PROP} \rangle, ?c) \text{ AND } (?q, \langle \text{ANNOT.PROP} \rangle, ?c) \quad (6.4)$$

The mapping Ω_r of relevant resources and their SKOS annotations is obtained by retrieving all resources q that share at least one SKOS concept c with resource r from AD (Eq. 6.5). The corresponding result set contains mappings for all variables (i.e., $?c$ and $?q$) in the triple

statements of P_r .

$$\Omega_r = [[P_r]]_{AD} \quad (6.5)$$

The SKOSRec engine extracts relevant resources and annotations from SPARQL endpoints. However, the OpenLink Virtuoso server enables definition of upper bounds for the number of records that can be retrieved with a single query [255]. Thus, SPARQL endpoint providers can control consumption of their resources. While this is a helpful feature on the provider's side, it requires the SKOSRec system to perform multiple queries. For *on-the-fly retrieval*, the engine needs to extract all resources with matching features to guarantee that the ranking can be properly conducted. Therefore, the system has to determine the number of records the endpoint holds for a given item. In this process, the input resource from the user profile (r) is excluded from retrieval (Listing 6.3).

Listing 6.3: Row query

```

1 SELECT (COUNT(DISTINCT ?q ?c) as ?count)
2 WHERE
3 {
4   VALUES ?c { <ANNOT(r)>}
5   ?q <ANNOT.PROP> ?c .
6   FILTER (?q != <r>)
7 }
8 GROUP BY ?q

```

After sending the row query to a SPARQL endpoint, the SKOSRec engine performs the before mentioned extraction of resources and annotations through multiple endpoint accesses, in case the result size exceeds the maximum number of rows specified by the provider. For this purpose, certain SPARQL expressions are applied to extract ordered slices of the final solution. One of these expressions is the `OFFSET` keyword. It marks the starting point of the intermediate solution from the entire sequence of solutions. The `LIMIT` expression is applied to restrict the result size to the defined maximum number of rows. Listing 6.4 presents the respective SPARQL query for ordered subset processing. This query follows the Virtuoso documentation for handling SPARQL endpoint constraints. The documentation also proposes the application of subqueries to circumvent sorting limitations [255].

Both row and multiple access queries are embedded in a well-defined procedure that subsequently extracts slices of the overall solution. This procedure iteratively increments the current offset point with the maximum row limit. Algorithm 1 lists the respective pseudocode.

Upon extraction of relevant resources and annotations, the process of similarity calculation can start. The calculation is based on the IC of the shared SKOS annotations of two resources. This approach is based on the hybrid similarity metric proposed by Meymandpour and Davis for LOD-enabled RS [165]. However, while their measure considers all types of IRI resources for

Listing 6.4: Relevant resource query (*simQuery*)

```

1 SELECT ?q ?c WHERE
2 {
3     VALUES ?c { <ANNOT(r)> }
4     ?q <ANNOT.PROP> ?c
5     {
6         SELECT ?q (COUNT(?subject) as ?c) WHERE
7         {
8             VALUES ?c { <ANNOT(r)> }
9             ?q <ANNOT.PROP> ?c .
10            FILTER (?q != <r>)
11        }
12        GROUP BY ?q
13        ORDER BY ASC (COUNT(?c))
14    }
15 }
16 OFFSET <CURRENT.OFFSET>
17 LIMIT <MAX.ROWS>

```

Algorithm 1 Multiple access procedure

```

1: function PROCESSRESULTS(rowQuery, simQuery, maxRows)
2:    $T \leftarrow \text{getTimes}(\text{rowQuery})$ 
3:    $\text{currentOffset} \leftarrow 1$ 
4:   Declare results
5:   Declare slice
6:   for  $t \in \{1, \dots, T\}$  do
7:      $\text{rowString} \leftarrow \text{OFFSET } \text{currentOffset } \text{LIMIT } \text{maxRows}$ 
8:      $\text{slice} \leftarrow \text{getSlice}(\text{simQuery} + \text{rowString})$ 
9:      $\text{results.add}(\text{slice})$ 
10:     $\text{currentOffset} \leftarrow \text{currentOffset} + \text{maxRows}$ 
11:   return results

```

the retrieval, the similarity metric in this thesis purely relies on SKOS annotations (Definition 11).

Definition 11 (SKOS similarity). *Let $\text{Annot}(r)$ be the set of SKOS features of resource r and $\text{Annot}(q)$ the set of SKOS features of resource q and $q \in \{\mu(?q) \mid \mu \in \Omega_r\}$, then their similarity can be derived from the IC of their shared concepts $C_{\text{shared}} = \text{Annot}(r) \cap \text{Annot}(q)$*

$$\text{sim}(r, q) = \text{IC}(C_{\text{shared}}) \quad (6.6)$$

The IC of a set of SKOS concepts is measured by the sum of the inverse logarithms of each concept's frequency in relation to the maximum frequency among all relevant resources (Definition 12).

Definition 12 (SKOS Information Content). *The SKOS IC is defined as an aggregation of the individual IC values of each concept that is contained in the set of shared features ($c \in C_{\text{shared}}$), where $\text{freq}(c)$ is the frequency of c among all relevant resources and n is the maximum frequency among these resources. The final similarity score of two resources r and q is determined*

by summing the IC values of each concept of the shared feature set.

$$IC(C_{shared}) = - \sum_{c \in C_{shared}} \log \left(\frac{freq(c)}{n} \right) \quad (6.7)$$

By considering the maximum frequency (n) instead of the number of LOD resources as was proposed by Meymandpour and Davis [165], it is ensured that IC scores have an upper bound. The determination of concept frequencies is straightforward. For this purpose, a SPARQL query (see Listing 6.5) is issued against the dataset.

Listing 6.5: Concept count

```

1 SELECT ?c (COUNT (DISTINCT ?x) as ?count)
2 WHERE
3 {
4   VALUES ?c {<Annot (?r)>}
5   ?x <ANNOT.PROP> ?c .
6 }
7 GROUP BY ?c

```

After having obtained resource annotations as well as IC values, similarity scores between the input resource r and each potentially relevant resource q can be calculated (Algorithm 2).

Algorithm 2 Computation of similarity scores for all relevant resources

```

1: function COMPUTEScores( $\Omega_r$ , Annot( $r$ ), ICValues)
2:   Declare icValue
3:   Declare scores
4:   for  $q \in \Omega_r$  do
5:     score  $\leftarrow$  0
6:     Annot( $q$ )  $\leftarrow$   $\Omega_r$ .getAnnotations( $q$ )
7:     for  $c \in$  Annot( $r$ ) do
8:       if Annot( $q$ ).contains( $c$ ) then
9:         icValue  $\leftarrow$  ICValues.get( $c$ )
10:        score  $\leftarrow$  score + icValue
11:      scores.put( $q$ , score)
12:   return scores

```

When a user profile contains more than a single item, similarity values are aggregated through summation to determine the final recommendation score (Definition 13).

Definition 13 (Recommendation score). *The recommendation score of a potentially relevant resource q is quantified by the sum of similarities with each resource r that can be found in the profile (Pr).*

$$score(Pr, q) = \sum_{r \in Pr} sim(r, q) \quad (6.8)$$

Upon calculation of similarity values for each potentially relevant item q , the engine ranks the retrieved resources based on a given limit (k) that is specified by the user (Algorithm 3).

The workflow steps are executed as successive SPARQL queries and the system processes the corresponding solutions to identify LOD resources that have the highest similarity with the

Algorithm 3 Extraction of k recommendations

```

1: function GETRECOMMENDATIONS(scores, k)
2:   Declare recommendations
3:   sortedScores = scores.sortByScoreDesc()
4:   for  $q \in \text{sortedScores}$  do
5:     if  $\text{recommendations.count}() < k$  then
6:       recommendations.add(q)
7:     else
8:       break
9:   return recommendations

```

user profile. It has to be ensured that favored items do not show up in the recommendation list (Algorithm 4, line 18). Additionally, the engine needs to execute all previously described workflow steps for each LOD resource in the profile (Pr) thereby determining a global list of recommendation scores based on the sum of similarity values for each resource q that has been extracted from the SPARQL endpoint (Algorithm 4, lines 5-23).

Algorithm 4 Recommendation procedure

```

1: Declare scores
2: Declare recommenderJob
3: Declare simEngine
4: function GETRECOMMENDATIONS( $Pr$ )
5:   for  $r \in Pr$  do
6:     Declare Annot(r)
7:     Declare ICcounts
8:     Declare ICvalues
9:     Declare  $\Omega_r$ 
10:    Declare localScores
11:    Declare recommendations
12:     $\text{Annot}(r) \leftarrow \text{recommenderJob.getAnnotations}(r)$ 
13:     $\text{ICcounts} \leftarrow \text{recommenderJob.getConceptCounts}(\text{Annot}(r))$ 
14:     $\text{ICvalues} \leftarrow \text{recommenderJob.getICscores}(\text{ICcounts})$ 
15:     $\Omega_r \leftarrow \text{recommenderJob.getRelevantResources}(r, \text{Annot}(r))$ 
16:     $\text{localScores} \leftarrow \text{simEngine.computeScores}(\Omega_r, \text{Annot}(r), \text{ICvalues})$ 
17:    for  $q \in \text{localScores}$  do
18:      if  $\text{!Pr.contains}(q)$  then
19:        if  $\text{Scores.contains}(q)$  then
20:           $\text{newScore} \leftarrow \text{scores.get}(q) + \text{localScores.get}(q)$ 
21:           $\text{scores.put}(q, \text{newScore})$ 
22:        else
23:           $\text{scores.put}(q, \text{localScores}(q))$ 
24:     $\text{recommendations} \leftarrow \text{simEngine.getRecommendations}(\text{scores}, k)$ 
25:    return recommendations

```

This chapter has presented key methods of *on-the-fly retrieval*, with which recommendations can be generated from LOD repositories through an API interface without requiring local metadata or excessive preprocessing operations other than the mapping of profile items with LOD resources.⁵

⁵Di Noia et al. describe, how a mapping procedure can be conducted for a real-world implementation. For their LDRS, they matched movie titles with the corresponding LOD resources in DBpedia. The authors extracted IRIs, labels and publication dates for all movie resources. They applied the Levenshtein distance on these resources to identify matching movies items [67].

6.4 Fast On-the-Fly Recommendations

Recommendation requests may encounter long processing times, especially when certain SKOS annotations are frequently used in a collection. Consider the DBpedia category `dbc:Living_People`, which in the 3.9 version of the dataset occurs in 49.6% of entities typed as music groups. To identify similar items for an input resource being annotated with this category, the engine would have to process hundreds of thousands of triple statements. Thus, the approach has to be complemented with an efficient technique that reduces the number of triple statements during runtime retrieval without corrupting the ranking. A possible solution is to check, which resources can be omitted without changing the order of results based on the number of recommendations (k). Therefore, an estimation query (Listing 6.6) retrieves all resources ($?q$) that have the same annotations ($?c$) as the input resource in descending order of matching annotations.

Listing 6.6: Estimation query

```

1 SELECT ?q (COUNT(DISTINCT ?c) as ?count)
2 WHERE
3 {
4   VALUES ?c { <ANNOT(r)> }
5   ?q <ANNOT.PROP> ?c .
6   FILTER (?q != <r>)
7 }
8 GROUP BY ?q
9 ORDER BY DESC (COUNT(?c))
10 OFFSET <CURRENT.OFFSET>
11 LIMIT <MAX.ROWS>

```

Upon extraction of the result set for the given query, the system initializes its optimization procedure. It obtains the number of matches (m) at the k^{th} position of the sorted result list. Thus, it is made sure that the final ranking is not corrupted by excluding too many resources from retrieval.

The number of shared annotations is subsequently decremented to test whether the sum of the m least informative item features is higher than the $m - 1$ most informative item features (Algorithm 5). If this is the case the engine does not have to retrieve LOD resources with less than m matching SKOS annotations, since this would not change the final ranking.

Algorithm 5 Computing the cut value

```

1: function COMPUTECUT( $m$ )
2:    $minScore \leftarrow getScore(m, false)$ 
3:    $cut \leftarrow m$ 
4:    $m \leftarrow m - 1$ 
5:   if  $m > 0$  then
6:      $maxScore \leftarrow getScore(m, true)$ 
7:     if  $maxScore \geq minScore$  then
8:        $cut \leftarrow computeCut(m)$ 
9:     else
10:      return  $cut$ 
11:   else
12:     return  $cut$ 

```

The process of similarity computation is imitated by calculating potential scores for all known subject annotations of the input resource (Algorithm 6). As soon as the maximum potential score for a certain number of shared features is smaller than the minimum score for a higher number of shared features, the *cut* value is set to m and the process of optimized similarity calculation can start.

Algorithm 6 Computing potential similarity scores

```

1:  $icValues.sort()$  ▷ Sort IC values in ascending order
2:  $size \leftarrow icValues.length()$ 
3: function GETSCORE( $m, max$ )
4:    $sum \leftarrow 0$ 
5:   if  $max == true$  then
6:      $sublist \leftarrow icValues.sublist((size - 1) - m, size - 1)$ 
7:   else
8:      $sublist \leftarrow icValues.sublist(0, m)$ 
9:   for  $icValue \in sublist$  do
10:     $sum \leftarrow sum + icValue$ 
11:  return  $sum$ 

```

The reduced result set is extracted by a SPARQL query that omits all resources which only share the number of subject annotations with the input resource that is equal to or below the threshold *cut* value. The remaining resources ($\Omega_{reduced}$) are joined with the non-optimized result set (Ω_r) (see 6.9 and 6.10) in the WHERE part of the query.

$$\Omega_{cut} = \{\mu|?q \mid \mu \in F_{count(?c)>cut}(\Omega_r)\} \quad (6.9)$$

$$\Omega_{reduced} = \Omega_r \bowtie \Omega_{cut} \quad (6.10)$$

The corresponding SPARQL query obtains $\Omega_{reduced}$ with an aggregated subquery containing the calculated *cut* value in the HAVING condition (Listing 6.7).

The optimized workflow is illustrated by Example 3 and Fig. 6.3. They describe the functioning of fast retrieval for an input movie resource from DBpedia.

Example 3. *A user stated that he likes the Western movie „They Call Me Trinity“ (dbr:They_Call_Me_Trinity) and would like to receive 20 suggestions for this preference (1). The SKORec engine extracts DBpedia categories from the SPARQL endpoint for this LOD resource via the property dct:subject (2). Additionally, IC scores are calculated based on category occurrences in the dataset (3). An estimation query identifies the starting point of the optimization process. In the case of the movie „They Call Me Trinity“ each resource sharing only two annotations with the input resources can be excluded from similarity calculation because there exist at least 20 movies with three matching annotations and even the three least informative concepts can achieve a higher similarity score than the two most informative concepts. Thus, the final ranking would not be changed by the exclusion of resources with fewer*

Listing 6.7: Query for optimized retrieval of relevant resources and annotations

```

1 SELECT ?q ?c
2 WHERE
3 {
4     VALUES ?c { <ANNOT(r)>}
5     ?q <ANNOT.PROP> ?c .
6     {
7         SELECT ?q (COUNT(?c) as ?count)
8             WHERE
9                 {
10                    VALUES ?c { <ANNOT(r)>}
11                    ?q <ANNOT.PROP> ?c.
12                }
13        GROUP BY ?q
14        HAVING (COUNT(?c)) >= <CUT>
15        ORDER BY ASC(COUNT(?c))
16    }
17 }
18 OFFSET <CURRENT.OFFSET>
19 LIMIT <MAX.ROWS>

```

annotations (5). Then, the engine calculates similarity values on a by 99.99 % reduced set of resources and annotations (6), based upon which the final ranking list is compiled (7).

This procedure is assumed to scale to large data spaces, i.e., LOD repositories with millions of annotated items and frequently occurring subject annotations, such as DBpedia [60] as it can considerably reduce the set of resources that have to be extracted from SPARQL endpoints (RQ10). Thus, the approach may decrease computational effort (RQ9).

6.5 Flexible Similarity Detection

6.5.1 Concept Expansion

While the ad-hoc retrieval approach is a useful feature of the SKOSRec engine, it does not take advantage of the semantic relations in the SKOS annotation graph, which may be helpful to explore RDF graphs more comprehensively (RQ3). In IR systems, KOS-based semantic expansion has a long tradition. Automatic query expansion (AQE) dates back to the 1960s. AQE addresses the problem of brief user queries and natural language ambiguity [49]. It is commonly applied to augment keywords with additional terms of similar meaning to improve retrieval results. Thus, document keywords that fit a user's underlying information need, but are not contained in the original query can still be found. KOSs have shown to be a viable tool for query expansion in IR systems [49]. Hence, it is no wonder that SKOS vocabularies have been recently adopted for concept expansion in RDF-based retrieval contexts as well [97, 226].

Because of the similarity of IR systems and CB RS, ideas of AQE approaches can be transferred to recommendation tasks to improve results and to help users better explore knowledge graphs. The authors of [67] and [66] provide evidence for this hypothesis. In their LDRS, they enhanced SKOS annotations with related concepts by following `skos:broader` links. The method im-

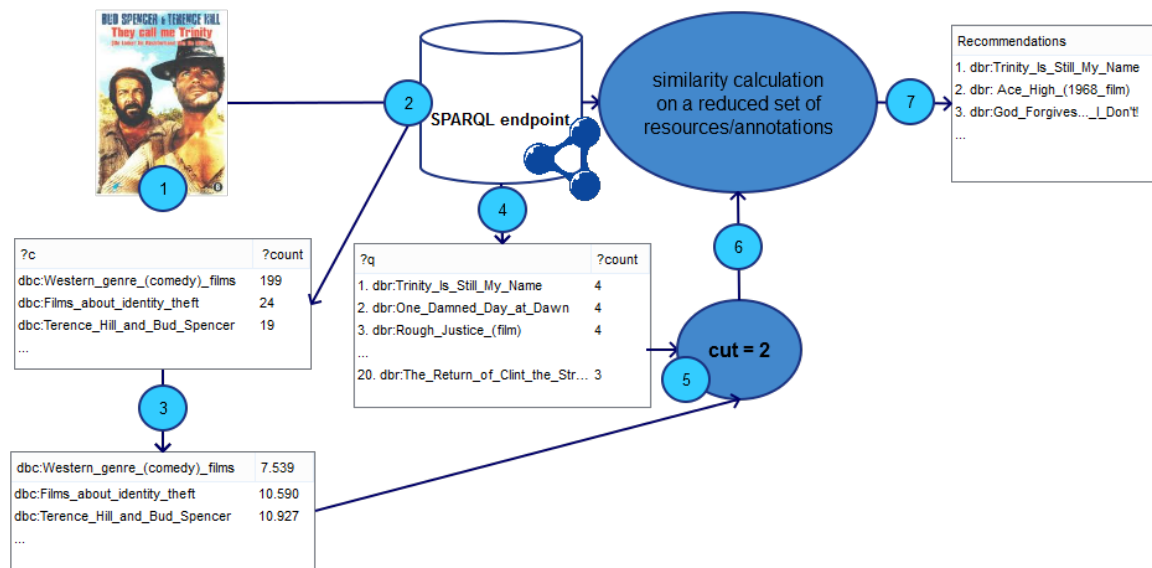


Fig. 6.3: Fast on-the-fly recommendations

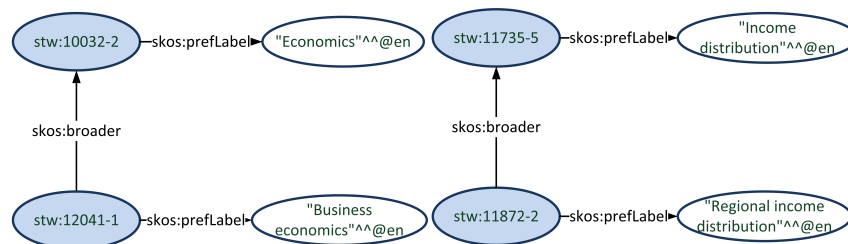


Fig. 6.4: Example subordinate/superior relations in the STW vocabulary

proved *accuracy* values. Additionally, Ruotsalo et al. quantified relatedness of annotations by a taxonomy-based similarity metric [208].

Based on the previous work, it is suggested to facilitate incorporation of taxonomically related concepts into the retrieval process of the SKOSRec engine. This approach might allow a more comprehensive exploration of the semantic space of the SKOS annotation graph. The author follows the idea of Ruotsalo et al., who assume that a quantified concept-to-concept value is better suited to reflect concept relatedness than an exploration of direct semantic relationships on `skos:broader` links. Consider the following example from the STW vocabulary: The SKOS pairs „Economics“/„Business economics“ and „Income distribution“/„Regional income distribution“ (Fig. 6.4) are connected by a `skos:broader` property indicating a subordinate/superior relation in the SKOS graph.

While in plain sight, it is evident that the SKOS concepts „Income distribution“/„Regional income distribution“ are more similar than the SKOS concepts „Economics“/„Business economics“, this difference is not detected, when only direct relations are considered because the

concepts' positions in the SKOS system are not taken into account. Therefore, concept-to-concept similarity scores are often applied to quantify relatedness in a taxonomy.

6.5.2 Concept-to-Concept Similarity

Concept-to-concept similarity metrics calculate scores based on the notion of the relative informativeness of a concept in a knowledge graph. Knowledge-based similarity metrics take advantage of the fact that most KOS systems are directed acyclic graphs (DAGs) or can at least be converted into a model that resembles a DAG [64, 187]. This structure evolves because domain-specific concepts tend to be organized as a tree-like structure with a root concept and multiple inheritances. Concepts are more specific the further down they are positioned in a taxonomy [64, 153]. These principles equally apply to Linked Data-enabled KOS. A qualitative analysis of numerous SKOS vocabularies revealed that most of these taxonomies define root concepts (`skos:hasTopConcept`) and do not contain cyclic hierarchical relationships [155].

Hence, general knowledge-based similarity metrics can be applied to SKOS vocabularies as well. The majority of these measures base concept-to-concept similarity calculation on the specificity of the concept(s) in question. Hence, the more specific two concepts are and the closer they are positioned in the taxonomy graph, the more similar they are.

The best-performing metrics compared to human judgment proved to be those that rely on the notion of the most informative common ancestor of two concepts v and z ($MICA_{v,z}$), i.e., the concept in the set of common ancestors of v and z ($CA_{v,z}$) that has the maximum informativeness value ($\theta(c)$) of all informativeness values ($\Theta_{v,z}$) of common ancestors (Eqs. 6.11 and 6.12) [30, 98, 202, 257].

$$\Theta_{v,z} = \{\theta(c) \mid c \in CA_{v,z}\} \quad (6.11)$$

$$MICA_{v,z} = \{c \mid \theta(c) = \max_{\theta(c) \in \Theta_{v,z}} \theta(c)\} \quad (6.12)$$

Since DAG-like taxonomies contain multiple inheritances, the cardinality of $MICA_{v,z}$ can be higher than one [30]. However, this does not represent a problem, as only the maximum informativeness value is needed for similarity calculation.

Concept informativeness can be calculated as a function of different parameters. It can be either derived from an external text corpus (extrinsic approach) or through exploration of the semantic structure of the taxonomy graph itself (intrinsic approach) [98]. When applying the intrinsic approach, specificity is often indicated by the depth of a concept in the hierarchy. In graph theory,

common ancestors with maximum depth are called *lowest common subsumers* [30, 146].

For instance, Wu and Palmer invented a concept-to-concept similarity metric based on concept depth ($\theta_{depth}(c)$) [257]. Because taxonomies typically have a root node, concept depth can be measured by the shortest path from the root to the concept [98]. Another intrinsic method to measure concept specificity is to determine the number of concepts that are subsumed by the concept in question ($Desc(c)$) in comparison to the overall number of concepts (C) that are contained in the taxonomy graph (Eq. 6.13) [221].

$$\theta_{desc}(c) = 1 - \frac{\log(Desc(c))}{\log(|C|)} \quad (6.13)$$

Concept-to-concept similarity values can be calculated based on the above-listed informativeness metrics. Often, this is done by setting the $MICA_{v,z}$ in relation to the sum of the informativeness of v and z (Eq. 6.14) [146, 257].

$$conceptSim(v, z) = \frac{\theta(MICA_{v,z})}{\theta(v) + \theta(z)} \quad (6.14)$$

The previous paragraphs have introduced approaches to concept-to-concept similarity calculation in taxonomy graphs. They have to be applied with caution on SKOS vocabularies since they only work well on transitive relationships. However, the SKOS specification does not require hierarchical relations (`skos:broader/skos:narrower`) to be transitive by default. Instead, data providers have to specify entailment explicitly through declaration of `skos:broader/skos:narrower` links as sub-properties of `skos:broaderTransitive/skos:narrowerTransitive` [24, 155, 166]. Thus, it is open to publishers to include entailment regimes, when necessary. Quality assessments of existing SKOS vocabularies have shown that only a few thesauri declare transitive relationships [24, 155]. Among the SKOS vocabularies of the usage scenarios, just the STW thesaurus has transitive connections, whereas in AGROVOC and the DBpedia category graph hierarchical links are simple relations. For this reason, the aforementioned concept-to-concept similarity metric can only be applied to the STW thesaurus. Transitivity is not the only problem when considering to use SKOS vocabularies for similarity-based concept expansion. For instance, quality assessments on the DBpedia category system have shown that the graph contains more than 1,000 cycles [155]. Therefore, Stankovic et al. introduced an alternative approach that can compute concept-to-concept similarities on the category graph. The corresponding metric is called *hyProximity* (*hyP*) [233]. In contrast to approaches that rely on the specificity of lowest common ancestors, it utilizes a modified path-based metric. This method goes back to Rada et al., who propose to take the shortest path length between two concepts as an indicator of similarity [199].

Stankovic et al. identify similar concepts for a set of initial seed concepts by traversing chains of `skos:broader` links. Concepts that are found within a shorter distance of the initial

seed concepts are more alike. Additionally, the authors assume that concepts appearing in the proximity of many seed concepts are more similar than those in the proximity of a single seed concept [233]. Hence, for a set of initial concepts and another concept, similarity is calculated as the sum of inverted shortest-path distances. Distance values ($d(v, z)$) are determined according to the first common ancestor when going along `skos:broader` relations. The *hyP* metric can be adapted to cases with only a single initial concept, such that it can be applied as an alternative concept-to-concept measure for concept expansion on cyclic vocabularies. This version of the metric (Eq. 6.15) will be used for the *flexible similarity detection* method of the SKOS-Rec engine on the DBpedia SKOS graph. The $conceptSim_{hyP}(v, z)$ metric does not consider lowest common ancestors because concept specificity cannot be unambiguously determined in a cyclic taxonomy.

$$conceptSim_{hyP}(v, z) = \frac{p(v, z)}{d(v, z)} \quad (6.15)$$

While it would be possible to delete circles in the SKOS graph, it might not be clear which relations could be omitted without inflicting information loss. A heuristic shortest-path-based approach is the only option to obtain semantically related concepts from a low-quality vocabulary. Therefore, Stankovic et al. introduce a pondering function to account at least a little for the idea of concept depth in the $conceptSim_{hyP}(v, z)$ metric (Eq. 6.16) [233]. The function decreases the final similarity score, the further away a common ancestor is positioned from two concepts.

$$p(v, z) = e^{-\gamma d(v, z)} \quad (6.16)$$

The following section will explore, how taxonomy-based similarity metrics can be applied for concept expansion in the context of *on-the-fly recommendation* retrieval.

6.5.3 Item Similarities with Concept Expansion

The SKOSRec engine performs concept expansion by identifying all concepts that are similar to the annotations of the LOD resource from the user profile (i.e., proximate concepts). The notion of proximate concepts ($Prox(c, r)$) is specified in Definition 14.

Definition 14 (Proximate concepts). *For a single SKOS annotation c of a user profile item, proximate concepts are those that are sufficiently similar to it, i.e., the SKOS-based concept-to-concept similarity score ($conceptSim(c, prox)$) exceeds a specified threshold ϵ (Eq. 6.17).*

$$Prox(c, r) = \{ prox \in AD \mid c \in Annot(r), conceptSim(c, prox) \geq \epsilon \} \quad (6.17)$$

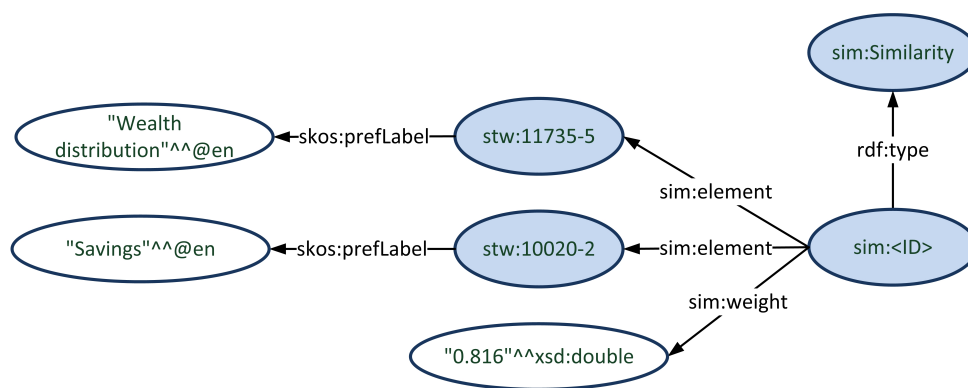


Fig. 6.5: RDF representation of a similarity object of two STW concepts

Concept-to-concept similarity values usually range between 0 and 1 with higher scores indicating higher proximity between concepts [35]. Favorably, the threshold level ϵ should be set reasonably high (i.e., with a minimum value of 0.5) to ensure that only relevant concepts are considered for similarity calculation. Additionally, since *on-the-fly retrieval* requires low computational costs, it is advisable to at least precompute concept-to-concept similarities, which can be saved in a local RDF triple store to be available at runtime. The data should adhere to LOD publishing principles and favorably be modeled according to a suitable LOD ontology in order to be conveniently processable by the SKOSRec engine. One such vocabulary is the *Music Ontology* by Jacobson et al. [120]. It was originally developed to store similarity values for music items, but can be applied for SKOS-based scores as well. Figure 6.5 shows an example similarity object. It stores the similarity values of two concepts from the SKOS-based STW Thesaurus.

When the local triple store contains concept-to-concept similarity values, they can be extracted with a SPARQL request during execution. Listing 6.8 shows the respective query for the RDF graph depicted in Figure 6.5.^{6,7}

Listing 6.8: Extraction of similar concepts

```

1 PREFIX sim: <http://purl.org/ontology/similarity/>
2
3 SELECT DISTINCT ?exact ?prox ?weight
4 WHERE
5 {
6   VALUES ?exact { <Annot(r)> }
7   ?id sim:element ?exact .
8   ?id sim:element ?prox .
9   ?id sim:weight ?weight .
10  FILTER (?weight >= <EPSILON>)
11  FILTER (str(?exact) != str(?prox))
12 };

```

⁶ PREFIX stw: <http://zbw.eu/stw/descriptor/>

⁷ PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

The request obtains all concepts that are similar to the annotations of the input resource. It also contains two SPARQL filter conditions. The first filter ensures that scores are above the specified concept-to-concept similarity threshold and the second filter excludes matching concepts. Formally, the set of all similar concepts for an input resource r from the user profile is the union of all proximate concept sets of the annotations (Eq. 6.18).

$$Prox(r) = \bigcup_{c \in Annot(r)} Prox(c, r) \quad (6.18)$$

Upon having extracted suitable concepts for expansion, the retrieval process for annotations and relevant resources is started with an enhanced set of annotations.

Definition 15 (Relevant resources and annotations (concept expansion)). *The expanded solution set comprises mappings for resources and their SKOS annotations that share at least an annotation with either the original annotations of the input resource (exact matches) (Ω_r) or with one of the previously identified similar concepts (proximate matches) ($c \in Prox(r)$). The processor evaluates the specified graph patterns and returns the mapping Ω_{exp} (see Eqs. 6.19 and 6.20).*

$$P_{exp} = (?q, <ANNOT.PROP>, ?c) \quad (6.19)$$

$$\Omega_{exp} = \{ \mu \in [[P_{exp}]]_{AD} \mid \mu(?c) \in Prox(r) \} \cup \Omega_r \quad (6.20)$$

After having obtained the result set, resources need to be ranked thereby considering proximate concepts as well. The following considerations drove the development of the scoring model.

- Exact matchings of LOD resources should have priority since they are the best indicators of item-level similarity. Thus, as long as resources share an annotation, proximate concepts are excluded.
- When two resources r and q do not share an annotation, the system should look for proximate concepts to make sure that all available information about resource similarity is taken into account. In case resource q is annotated with more than a single concept that is similar to the original annotation in question; the maximum score is selected.

The importance of a single annotation c (i.e., descriptor score), when calculating the similarity score of two resources r and q , is determined by both the IC of the respective annotation and the similarity score between annotation c and annotations of q . The notion of the descriptor score is specified in Definition 16.

Definition 16 (Descriptor score). *When an exact annotation is present in the set of annotations of q , its score is represented by the IC of c . If this is not the case and c has similar concepts that are present in the annotation set of q ; the concept-to-concept similarity value is utilized to decrease the IC. Hence, the respective annotation is contributing slightly less to the final similarity score of the two resources. When an annotation neither appears in the set of annotations of q nor has similar concepts that are present in the annotation set of q , the descriptor score is set to 0 (Eq. 6.21).*

$$descScore(c, r, q) = \begin{cases} IC(c), & c \in Annot(q) \\ simMAX(c, r, q) \cdot IC(c), & c \notin Annot(q), Scores(c, r, q) \neq \emptyset \\ 0, & otherwise \end{cases} \quad (6.21)$$

The engine identifies expansion candidates for an annotation ($c \in Annot(r)$) by looking at the annotations of a resource q , whose similarity score with r needs to be determined. The final concept-to-concept similarity value ($simMAX(c, r, q)$) is based on the scores of proximate descriptors (Definition 17).

Definition 17 (Concept-to-concept similarity score). *When a proximate concept ($prox$) appears in the set of similar concepts of the input annotation ($Prox(c, r)$) as well as in the annotations of q ($Annot(q)$), its concept-to-concept similarity score is added to the set of relevant values (Eq. 6.22). From these values, the maximum similarity score is selected as the concept-to-concept similarity score (Eq. 6.23).*

$$Scores(c, r, q) = \{ conceptSim(c, prox) \mid prox \in Prox(c, r) \cap Annot(q) \} \quad (6.22)$$

$$simMAX(c, r, q) = \max_{s \in Scores(c, r, q)} Scores(c, r, q) \quad (6.23)$$

The recommendation score of two items r and q in concept expansion mode is specified in Definition 18

Definition 18 (Recommendation score (concept expansion)). *The scores of each concept of the annotation set of r are aggregated by summation to determine the final similarity value of two resources r and q in concept expansion mode.*

$$score_{exp}(r, q) = \sum_{c \in Annot(r)} descScore(c, r, q) \quad (6.24)$$

Algorithmically, this procedure does not differ much from the computation of similarity scores in the execution mode of *on-the-fly retrieval*. The only difference is that for each LOD resource $q \in \Omega_{exp}$ the engine takes into account the concept-to-concept similarity scores of related concepts as well (see Algorithm 7).

Algorithm 7 Computation of recommendation scores (concept expansion)

```

1: function COMPUTESCORES( $\Omega_r$ ,  $Annot(r)$ ,  $simConcepts$ ,  $ICValues$ ,  $\Omega_{exp}$ )
2:   Declare  $scores$ 
3:   Declare  $icValue$ 
4:   for  $q \in \Omega_{exp}$  do
5:      $score(r, q) \leftarrow 0$ 
6:      $Annot(q) \leftarrow \Omega_{exp}.getAnnotations(q)$ 
7:     for  $c \in Annot(r)$  do
8:       if  $Annot(q).contains(c)$  then
9:          $icValue \leftarrow ICValues.get(c)$ 
10:         $score(r, q) \leftarrow score(r, q) + icValue$ 
11:       else
12:          $simMAX(c, r, q) \leftarrow 0$ 
13:          $Prox(c) \leftarrow simConcepts.getSimilarConcepts(c)$ 
14:         for  $p \in Prox(c)$  do
15:           if  $Annot(q).contains(p)$  then
16:             if  $simConcepts.getSimilarity(c, p) > simMAX(c, r, q)$  then
17:                $simMAX(c, r, q) \leftarrow simConcepts.getSimilarity(c, p)$ 
18:              $score(r, q) \leftarrow score(r, q) + icValue \cdot simMAX(c, r, q)$ 
19:          $scores.put(q, score(r, q))$ 
20:   return  $scores$ 

```

Consider Example 4 that is depicted in Figure 6.6. It illustrates a *flexible similarity detection* procedure for the usage scenario of DL search in the LOD repository EconStor.

Example 4. A user has accessed the paper `econstor:21555`⁸ and would like to receive recommendations through flexible similarity detection (i.e., concept expansion). Therefore, he specifies a similarity threshold ($\epsilon > 0.75$). Hence, even though the input paper (`econstor:21555`) does not share the SKOS concept „Wealth distribution“ (`stw:11751-0`) with the LOD resource (`econstor:34662`), the related concept „Savings“ is incorporated into the retrieval process. This is because the precalculated concept-to-concept similarity value for „Wealth distribution“ and „Savings“ is sufficiently high ($conceptSim = 0.816$). The concept „Overlapping generations“ (`stw:10179-0`) is considered to be related as well but is not taken into account, because its similarity with the concept „Wealth distribution“ is smaller than the similarity value of the „Savings“ concept. Additionally, the fact that the SKOS concepts „Wealth distribution“ and „Savings“ do not match exactly is modeled through multiplication of the IC value of „Wealth distribution“ with the concept similarity score of „Wealth distribution“ and „Savings“ ($0.816 \cdot 0.7131$). Thus, related concepts contribute less to the final similarity score of two resources than matching concepts.

By application of the *flexible similarity detection* method, it may be possible to change the composition of result lists thus enabling users to explore RDF graphs more comprehensively.

⁸ PREFIX `econstor:` <<http://linkeddata.econstor.eu/beta/page/publications/>>

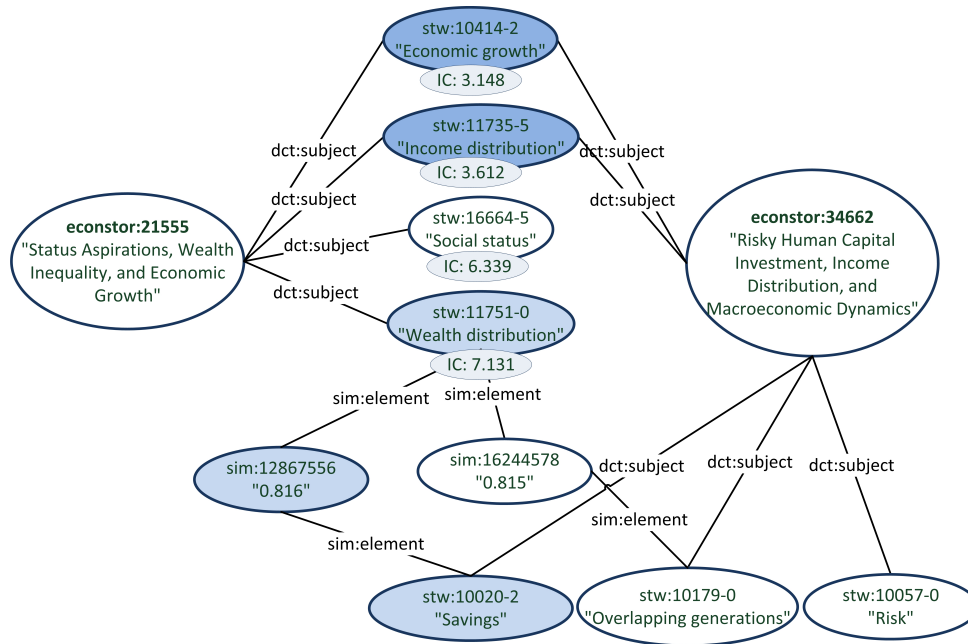


Fig. 6.6: Similarity detection with concept expansion

6.6 Constraint-based Recommendations

6.6.1 Prefiltering

To facilitate exploratory search in LOD repositories, users should also have the option to formulate conditions in their queries. A possible way to enable filters during retrieval is before similarity calculation. Thus, users can adjust recommendation lists according to their needs. Due to the expressiveness of RDF data, filter conditions cannot only be applied on attributes that are directly connected to potentially relevant items but can also be declared as a graph pattern (P). By this means, it is possible to retrieve LOD resources that are similar to the user profile and match an advanced condition. Before the extraction of relevant resources and annotations, the recommendation engine identifies the resources that satisfy the filter (Definition 19).

Definition 19 (Prefiltered resources). *The mappings of filtered resources $\Omega_{prefilter}$ are obtained by matching a specified graph pattern (P) to the RDF graph of a dataset D (Eq. 6.25).*

$$\Omega_{prefilter} = \{ \mu|_{?q} \mid \mu \in [[P]]_D \} \quad (6.25)$$

After pre-selection, resources are brought together with user profile data to retrieve the final set of recommendations (Definition 20).

Definition 20 (Annotations and relevant resources (prefiltered retrieval)). *The solution mappings of annotations and relevant resources in prefiltered retrieval mode are obtained by joining the set of prefiltered resources with the mappings Ω_r of all relevant resources and annotations as determined from the user profile.*

$$\Omega_{pre} = \Omega_{prefilter} \bowtie \Omega_r \quad (6.26)$$

The SPARQL query that is needed to perform the previously mentioned operations is given in Listing 6.9. Note that, in the provided query example, the user filter is contained in the graph pattern P (Listing 6.9, line 10) and can itself comprise several triple statements or subgroups of graph patterns. The grammar of the SKOSRec query language (Sect. 6.8) defines the allowed expressions for P in the `RecGroupGraphPattern` construct.

Listing 6.9: Query for *prefiltered constraint-based recommendation* retrieval

```

1 SELECT ?q ?c
2 WHERE
3 {
4     VALUES ?c { <ANNOT(r)> }
5     ?q <ANNOT.PROP> ?c .
6     {
7         SELECT ?q (COUNT(?c) as ?count)
8         WHERE
9         {
10            <P>
11            VALUES ?c { <ANNOT(r)> }
12            ?q <ANNOT.PROP> ?c .
13        }
14        GROUP BY ?q
15        HAVING (COUNT(?c)) >= <CUT>
16        ORDER BY ASC(COUNT(?c))
17    }
18 }
19 OFFSET <CURRENT.OFFSET>
20 LIMIT <MAX.ROWS>

```

Additionally, the system must ensure that the graph pattern contains a variable declaration that resembles the variable for potentially relevant resources in the surrounding query (i.e., $?q$). The compiler of the SKOSRec engine performs this consistency check. After retrieving annotations and relevant resources in *prefiltering* mode, the system generates *constraint-based recommendations*. In this context, the ranking procedure is adapted to the restricted set of relevant resources and annotations. Hence, IC values of matching annotations and respective SKOS similarities are quantified according to the user filter (Definitions 21 and 22). By this means, recommendation scores reflect the similarity of items for the actual set of filtered LOD resources.

Definition 21 (Conditional SKOS similarity). *Let $Annot(r)$ be the set of SKOS features of r and $Annot(q)$ the set of SKOS features of resource q , which is contained in the filtered set of*

annotations and relevant resources ($q \in \{\mu(?q) \mid \mu \in \Omega_{pre}\}$), then the similarity can be derived from the IC of their shared concepts ($C_{cond} = Annot(r) \cap Annot(q)$) (Eq. 6.27).

$$sim_{cond}(r, q) = IC(C_{cond}) \quad (6.27)$$

Definition 22 (Conditional SKOS Information Content). *The conditional IC of a set of SKOS annotations is defined by the sum of the IC of each concept $c \in \{\mu(?c) \mid \mu \in \Omega_{pre}\}$, where $freq_{cond}(c)$ is the frequency of c among all filtered resources and n is the maximum frequency among these resources (Eq. 6.28).*

$$IC(C_{cond}) = - \sum_{c \in C_{cond}} \log \left(\frac{freq_{cond}(c)}{n} \right) \quad (6.28)$$

The system carries out the calculations (i.e., determination of similarity values, the ranking of LOD resources) in the same way as in non-filtering mode (Sect. 6.3). The strengths of the approach are illustrated in the context of the travel usage scenario for a place resource from DBpedia. Example 5 and Fig. 6.7 show the effects of filtering.

Example 5. *A user profile contains a preference for the Lake Baikal (`dbr:Lake_Baikal`) region (e.g., as indicated by a positive rating on a travel community site). With simple on-the-fly retrieval, the user would receive a recommendation list that primarily consists of travel destinations that are similar to this geographical area. However, a filter facilitates more specific recommendations. In the example, the user could require suggestions for travel destinations that are known to be nature reserves, which changes the ranking of recommendation results accordingly.*

In the previous example, the filter is only applied to direct attributes of travel destinations. However, as has been argued before, the RDF data model enables more *expressive constraints*. For instance, when users formulate filters imprecisely, a graph pattern could enhance the query to match more attributes. Furthermore, *expressive filter* requests can address data quality issues by inferring additional triple statements. Simple filters, on the other hand, have a limited scope. In the example, they lead to recommendations for travel destinations that are located in proximity to the location specified in the user profile. In contrast, Example 6 and Figure 6.8 illustrate how graph patterns can help when a user has stated a preference for a destination in one region but intends to travel to a different geographical area.

Example 6. *The profile contains a preference for the Lake Baikal region. Additionally, the user specifies that he is interested in Southeast Asian travel destinations. With only a simple filter, the engine would not be able to generate a recommendation, since the resource `dbr:Lake_`*

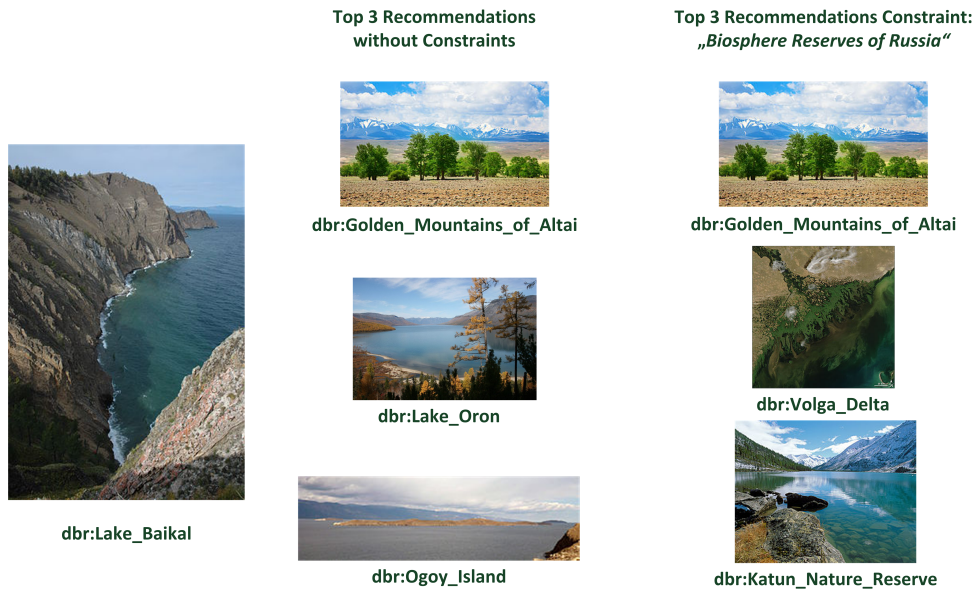


Fig. 6.7: *On-the-fly* vs. *constraint-based retrieval* (travel domain)

Baikal is not connected to any destination with the annotation `dbc:Southeast_Asia`. The system can only identify relevant items for this filter condition when it queries a couple of additional triple statements (see green nodes in Fig. 6.8). Similarity calculation is facilitated by matching SKOS annotations (as is shown by the blue node in Fig. 6.8). The remaining SKOS annotations in the figure are either connected to the DBpedia resource `dbr:Lake_Baikal` or the resource `dbr:Tonlé_Sap`. The concepts do not have IC scores (IC: NaN) because they do not appear in the set of filtered relevant resources and annotations.

Example 6 has shown how *expressive filtering* can help to make the most of the available data sources in LOD repositories. Thus, by specifying graph-based filter conditions, users may be assisted in finding relevant recommendations that fit their interests (RQ13).

6.6.2 Preference Querying

In addition to *prefiltering*, the engine can apply graph pattern matching to generate entire user profiles. In the previous sections, it was assumed that users express their tastes in the form of preference statements for concrete items. However, it may also be the case that likings are only vaguely known, for instance, when a user has not yet interacted with an online platform but can just provide general information while creating a profile (e.g., regarding genre preferences for movies). Another example is when users prefer products with specific features but are not able or willing to specify the actual items. For instance, consumers are often fans of particular movie directors or book authors. However, on popular online platforms, such as Amazon [11] or Netflix [172] common retrieval strategies either search items by their creators or obtain recommendations from past consumption behavior. A novel and potentially interesting search paradigm

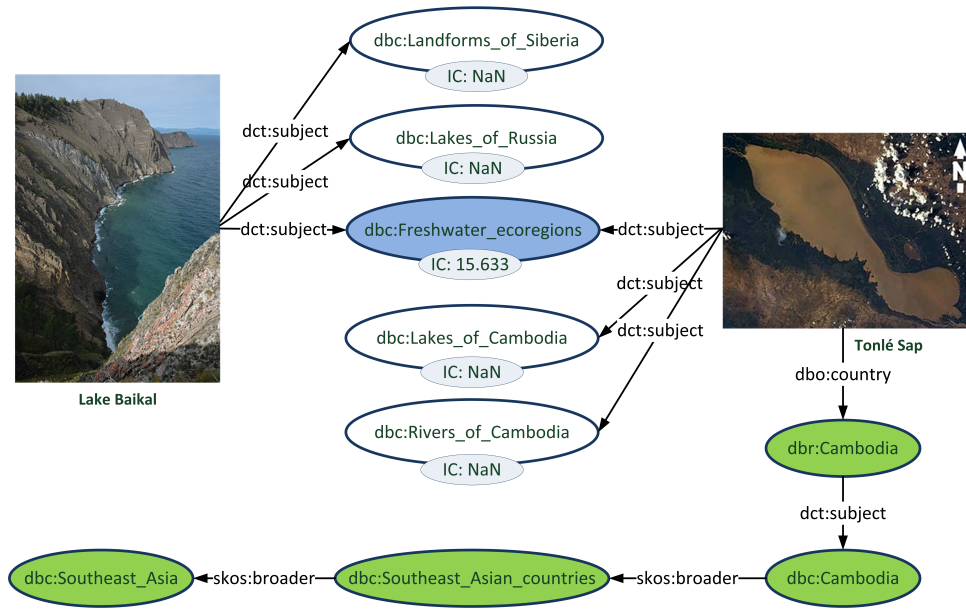


Fig. 6.8: *Constraint-based retrieval with graph pattern matching (travel domain)*

would be to combine both methods in a unified approach. With this approach, a profile is created based on a single user statement, upon which the engine extracts the required information from the LOD repository. This method is equally applicable to databases. However, they might lack persistent authority control (see Sect. 2.3), which, in turn, hinders extraction of all the relevant data for a particular entity. On top of that, the data may not be available in the local database, in which case LOD repositories can offer valuable additional information sources. Therefore, the SKOSRec engine can use graph-based queries for profile generation from RDF collections.

Definition 23 (Queried profile). *A queried profile is a user profile that has been compiled by retrieving all LOD resources r that are part of an annotation graph AD and are contained in the solution mappings resulting from the evaluation of a specified graph pattern (P_{pref}) over a dataset D (Eq. 6.29).*

$$Pr = \{ \mu(?r) \mid \mu \in [[P_{pref}]]_D, ?r \in \text{dom}(\mu), r \in AD \} \quad (6.29)$$

The engine can obtain such a profile by issuing a SPARQL query that contains the preference graph pattern (P_{pref}) in the WHERE condition (Listing 6.10).

In this case, the compiler has to make sure that P_{pref} contains the variable declaration (i.e., $?r$) that denotes the LOD resources to be extracted for the user profile. Regardless whether the user profile has been obtained through querying or by explicit preferences for specific items, it consists of a set of LOD resources that can be applied for recommendation retrieval. Example 7 illustrates a typical scenario for profile querying.

Listing 6.10: Preference query

```

1 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2 PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
3
4 SELECT ?r
5 WHERE
6 {
7   <P_pref>
8   ?r <ANNOT.PROP> ?c .
9   ?c rdf:type skos:Concept .
10 }

```

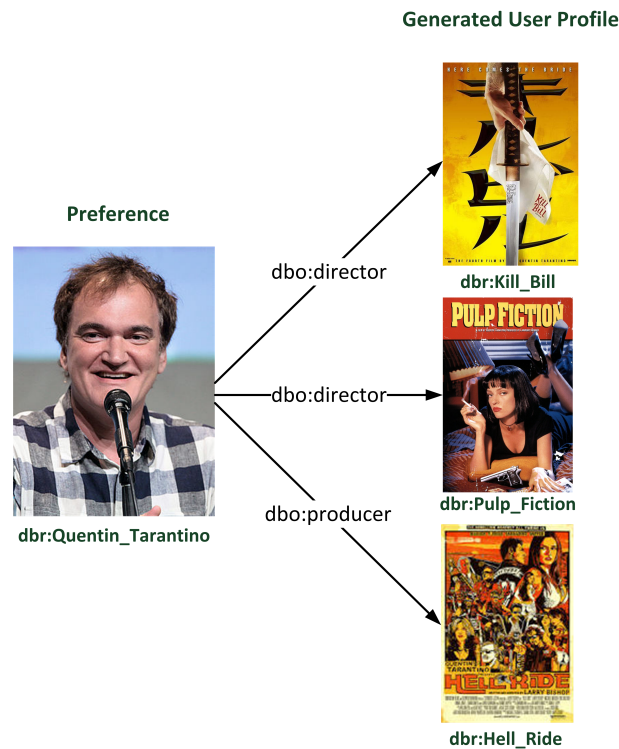


Fig. 6.9: Preference querying (movie domain)

Example 7. Suppose a user is interested in *Quentin Tarantino* movies, but does not specify the precise items he likes. He could send a preference query to the SKOSRec engine. The graph pattern in the request would match all movies that are connected to *dbr:Quentin_Tarantino* with the properties *dbo:director* or *dbo:producer* (see Fig. 6.9) and add them to the user profile.

6.6.3 Postfiltering

The SKOSRec engine cannot only combine similarity calculation and graph pattern matching to *prefilter* relevant LOD resources, it can also apply user constraints after similar items have already been determined. By these means, recommendations are filtered ex-post to the similarity detection process. This feature enables novel retrieval patterns, in which suggestions are part

of a subquery. Definition 24 specifies the notion of SKOSRec recommendations that undergo *postfiltering*.

Definition 24 (SKOSRec recommendations). *SKOSRec recommendations ($R(Pr)$) are the set of suggestions generated by processing the solution mappings (Ω .) obtained from recommendation retrieval. They can be a result of simple on-the-fly retrieval, flexible similarity detection, prefiltering or a result of a combination of these techniques. Pr represents the input profile that contains at least one preference for an item. The cardinality of $R(Pr)$ is the intended number of recommendations (k) that is specified by the user, based on which the engine selects the resources with the highest recommendation scores that are not contained in the profile (Eq. 6.30).*

$$R(Pr) = \{x \mid x \in \{q \mid \text{score}(Pr, q) \geq k \max_{q \in \Omega} \text{score}(Pr, q), q \notin Pr\}\} \quad (6.30)$$

The approach of subquerying with recommendation results builds on the idea that SKOSRec suggestions $R(Pr)$ can be joined with any graph pattern P_{post} past to the process of similarity calculation (Definition 25). It does not make a difference how the engine obtained these recommendations.

Definition 25 (Postfiltered recommendation mapping). *The postfiltered recommendation mapping is obtained by joining solution mappings generated from SKOSRec recommendations with the results of matching a postfilter graph pattern (Ω_{post}) (Eqs. 6.31 and 6.32).*

$$\Omega_{postfilter} = \{\mu \mid \mu \in [[P_{post}]]_D, ?x \in \text{dom}(\mu)\} \quad (6.31)$$

$$\Omega_{post} = \{\mu_{|?x} \mid x \in R(Pr)\} \bowtie \Omega_{postfilter} \quad (6.32)$$

As the *prefilter*, P_{post} can be comprised of triple statements that are eligible in the `RecGroup-GraphPattern` part of the SKOSRec grammar. The only precondition is that the recommendation variable (e.g., $?x$) is contained in the graph pattern to facilitate join operations. The compiler has to check this condition during recommendation retrieval. The SPARQL query in Listing 6.11 shows how a *postfiltered recommendation* mapping can be extracted from a LOD repository.

The set of projection variables of the corresponding SKOSRec query must contain the recommendation variable, because the final solution is ranked according to the similarity score of the suggestions. In case, users additionally specify a `LIMIT` clause, it refers to the total number of records to be retrieved for the recommendation request (see Sect. 6.8). When no `LIMIT` is

Listing 6.11: Query for *postfiltered recommendation* mapping

```

1 SELECT *
2 WHERE
3 {
4     VALUES ?x { <R(Pr)> }
5     <P_post>
6 }

```

specified, it may be that individual recommended items are linked to many other LOD resources via the specified P_{post} graph pattern. Hence, the size of the result set can become unpredictably large, which has to be handled by an appropriate output interface.

The complementary case of a single LOD resource being linked to numerous suggested items requires a different approach to *postfiltered retrieval*. For instance, a user would like to receive suggestions for a superordinate item that is based on preferences for sublevel entities (e.g., when generating travel recommendations for cities based on the POIs, a user visited in another city). These cases of aggregation-based retrieval have to be treated differently than a regular *postfilter request*. Apart from excluding sublevel items of the profile from the set of similar resources, the engine should also omit any connections between the preferred superordinate item (a) and similar sublevel entities. Additionally, the user has to specify the variable in his profile (e.g., $?y$) that represents the aggregation-based suggestions within the postfilter graph pattern P_{post} (Definition 26).

Definition 26 (Resources for aggregation-based retrieval). *The set of resources for aggregation-based retrieval is obtained by retrieving LOD resources that have a connection to a previously generated suggestion thereby excluding all resources that are linked to the specified superordinate item a in the profile (Eq. 6.33).*

$$R_{agg}(a) = \{\mu(?y) \mid \mu \in \Omega_{post}, ?y \in dom(\mu), a \notin \mu(?y)\} \quad (6.33)$$

For each potential aggregation-based suggestion (y), similarity scores of connected entities (x) have to be considered for the final ranking. In this context, different approaches to aggregation can be applied. In cases when both similar items and aggregation-based recommendations are entities on comparable levels of granularity (e.g., as in *cross-domain requests*, see Sect. 6.8), then choosing the maximum similarity score might be the best way to represent the relatedness of the resource to the user profile. On the other hand, when *postfiltered recommendations* are based on similarity scores of their sublevel entities (e.g., as in *rollup requests*, see Example 8), the scores of related suggestions should be aggregated by the sum or the average of individual scores.

Definition 27 (Aggregation-based ranking score). *The ranking score of an LOD resource y that is connected to a set of recommended items is determined by aggregating scores of connected*

entities either by the maximum (Eq. 6.34), the sum (Eq. 6.35) or the average (Eq. 6.36) of individual scores.

$$score_{aggMAX}(y) = \max_{x \in \mu(?x,y)} score(Pr, x) \quad (6.34)$$

$$score_{aggSUM}(y) = \sum_{x \in \mu(?x,y)} score(Pr, x) \quad (6.35)$$

$$score_{aggAVG}(y) = \frac{\sum_{x \in \mu(?x,y)} score(Pr, x)}{|\{x \in \mu(?x,y)\}|} \quad (6.36)$$

In the case of the aggregation-based retrieval mode, the engine makes sure that the corresponding SKOSRec query contains the aggregation variable in the *postfilter* section of the request because the final result set is ranked according to the aggregated score. If users specify a `LIMIT` clause in this section as well, the number of aggregation-based recommendations is restricted to this limit (see Sect. 6.8).

Example 8 illustrates the previous statements.

Example 8. *Suppose a user went on a city trip to London. He visited and liked certain POIs (e.g., Oxford Street) there. The user states these preferences in his profile and issues a corresponding request, based upon which the engine executes a fixed recommendation workflow. First, the system determines other similar POIs and extracts cities (`rdf:type yago:UrbanArea108675967`⁹) that are connected to these places in the LOD repository via subproperties of the property `dul:hasLocation`. An additional condition in the *postfilter* section of the query requires urban area LOD resources. During the aggregation of similarity scores, the engine omits sublevel entities that are connected to preference information in the user profile. Hence, London-based POIs are excluded from retrieval. The more relevant POIs a city shares with the POIs of the input profile, the higher it is ranked. Hence, in the example, Oxford is assumed to be more relevant, since it has more in common with the user profile than Lechlade.*

6.7 Cross-Repository Recommendations

The SKOSRec engine enables another type of retrieval pattern. It can obtain recommendations from distributed repositories, as mapping relations in SKOS vocabularies facilitate *cross-repository* requests. For this purpose, the engine utilizes the property `skos:exactMatch`.

⁹ PREFIX `yago:` <<http://dbpedia.org/class/yago/>>

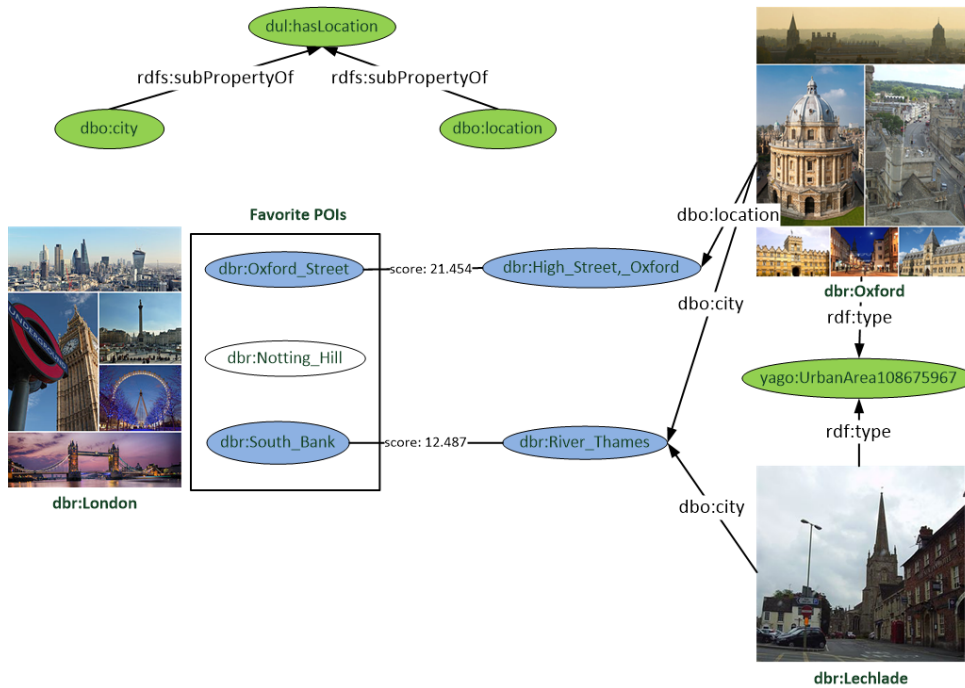


Fig. 6.10: Postfiltered recommendation (aggregation-based)

The property specifies concordance links between descriptors in different SKOS concept schemes (see Sect. 6.1). A `skos:exactMatch` relation indicates that two concepts share the same semantic meaning. For instance, in the Standard Thesaurus for Economics the concept `stw:12945-3` („Agriculture“) points to a corresponding descriptor in the AGROVOC thesaurus (`agrovoc:c_203`¹⁰). Since the approach of *on-the-fly retrieval* is based on concept annotations in a data collection, mapping links can facilitate detection of related resources from other repositories. The SKOS version of the STW thesaurus is used to describe publications from EconStor LOD [173], while AGROVOC descriptors are applied in the AGRIS LOD repository to annotate publications from the field of agriculture and agricultural economics [12]. Based on a user profile that contains preference information for items from a default repository that is stated in the standard configuration of the SKOSRec engine (e.g., EconStor LOD), the engine determines the annotated SKOS concepts of these items and looks for mappings to another vocabulary (e.g., AGROVOC). For the mapped concepts, potentially relevant items are determined from a target repository (e.g., AGRIS). Thus, the processing step of similarity calculation is based on the target repository’s SKOS annotations. Hence, any *pre-* or *postfilter* condition needs to be defined according to the specificities of the target repository.

Cross-repository recommendations depend on the existence of mapping relations between two SKOS vocabularies. When this precondition is met, the SKOSRec engine generates recommendations from distributed collections such that repositories are virtually integrated during retrieval. In this way, the system can handle the decentralized nature of the LOD cloud with-

¹⁰ PREFIX `agrovoc:` <<http://aims.fao.org/aos/agrovoc/>>

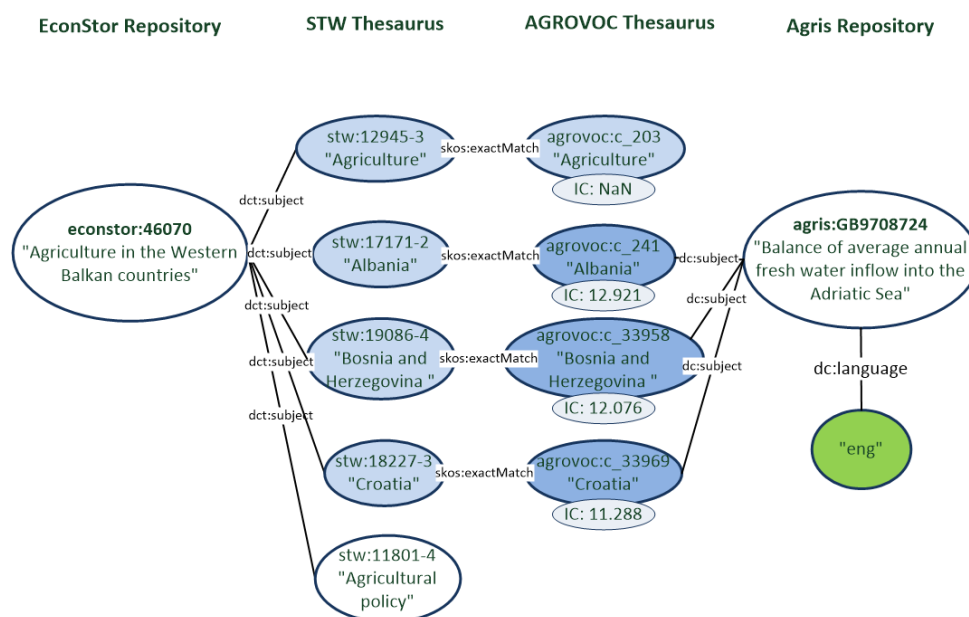


Fig. 6.11: Cross-repository retrieval

out requiring additional data integration tasks (RQ8). Example 9 describes a recommendation procedure for a particular request.

Example 9. *An economics researcher is interested in environmental and agricultural issues in the geographical region of Southeast Europe. The publication „Agriculture in the Western Balkan countries“ (`econstor:46070`) can be found in his user profile. The user is already familiar with all related publications in EconStor but would like to determine whether the AGRIS repository holds additional interesting material on this topic. However, he is only interested in English publications (`dc:language = 'en'`). For this query, the engine retrieves all SKOS annotations of `econstor:46070`, determines the matches of these annotations with the AGROVOC thesaurus and looks for potentially relevant publications in the AGRIS repository. Based on the specified filter condition (`dc:language = 'en'`) and the AGROVOC descriptors, the engine generates suggestions from the AGRIS repository (e.g., `agris:GB9708724`) and presents it to the user. Figure 6.11 illustrates this showcase scenario graphically.*

The example has shown how *cross-repository retrieval* can be carried out over LOD datasets. It is a straightforward approach that takes advantage of the knowledge structures of SKOS vocabularies. By these means, users are enabled to obtain recommendations from a target repository which are based on a user profile that only contains items from a source repository.

6.8 The SKOS Recommender Query Language

6.8.1 Syntax

The previous section described different ad-hoc retrieval techniques for LOD repositories. They build the technical foundation of the SKOSRecommender query language. The language facilitates the formulation of recommendation requests. While there already exist language extensions that combine SPARQL with similarity-based retrieval, these systems have some shortcomings that prevent full exploitation of the potential of LOD [23, 130]. The major drawback of these languages is that they do not facilitate *advanced queries* that execute recommendation workflows. Such a workflow is a precisely defined sequence of operations, of which a single processing step either involves similarity calculation or query-based retrieval. The goal is to generate personalized suggestions that provide an added-value to regular recommendations resulting from simple content-based retrieval. While other researchers have already explored the positive impact of recommendation workflows in the context of SQL-based RS [140], a powerful LOD-enabled query language has yet to be developed. Apart from this aspect, it has to be made sure that the new language correctly extends and combines elements of the latest SPARQL specification (i.e., SPARQL 1.1) and offers language constructs to express individual preferences (RQ4 and RQ5). Additionally, the syntax needs to provide statements that facilitate combinations of central system features of the SKOSRec engine, such as *flexible similarity calculation* or *constraint-based queries* in order to enable *advanced recommendation requests* (RQ6). Listing 6.12 shows the syntax of the SKOSRec query language. Each integral part of the language will now be explained in detail. In the given grammar, underlined parts represent SPARQL syntax elements that have been directly taken from the latest W3C specification [102]. Expressions in capital letters denote query keywords.

Listing 6.12: Grammar of the SKOSRec query language

```

1 SKOSRecQuery          ::= Prologue SelectPart? SimProjection PREF ItemPart+ RecWhereClause?
2 SelectPart           ::= SELECT (DISTINCT | REDUCED)? Var+ RecWhereClause
3                       ::= LimitClause? Aggregation?
4 Aggregation         ::= AGG IRIREF Var (SUM | MAX | AVG )
5 SimProjection       ::= RECOMMEND Var TOP INTEGER ServiceIntegration?
6 ServiceIntegration  ::= FROM SERVICE IRIREF
7 ItemPart            ::= (VarPart | IRI) Sim?
8 VarPart             ::= [ Var RecWhereClause ]
9 Sim                 ::= SIM Relation DECIMAL
10 Relation           ::= ( > | >= | = )
11 RecWhereClause     ::= WHERE RecGroupGraphPattern
12 RecGroupGraphPattern ::= '{' RecGroupGraphPatternSub '}'
13 RecGroupGraphPatternSub ::= TriplesBlock? (RecGraphPatternNotTriples '.' ? TriplesBlock)*
14 RecGraphPatternNotTriples ::= RecGroupOrUnionGraphPattern | RecOptionalGraphPattern |
15 RecMinusGraphPattern | Filter | Bind | InlineData
16 RecGroupOrUnionGraphPattern ::= RecGroupGraphPattern ( 'UNION' RecGroupGraphPattern)*
17 RecOptionalGraphPattern ::= 'OPTIONAL' RecGroupGraphPattern
18 RecMinusGraphPattern ::= 'MINUS' RecGroupGraphPattern

```

SKOSRecQuery (line 1) As a regular SPARQL `SELECT` query, a SKOSRec query always generates a table-like result set that contains at least a mapping to a single variable. A SKOSRec query can be comprised of up to six elements, of which the parts *SimProjection* and *ItemPart* are obligatory. Hence, users need to state at least their preferences and how many recommendations they would like to receive. Advanced retrieval patterns can be formulated as well. For instance, similar resource retrieval can be combined with *prefiltering* (*RecWhereClause*) and/or *postfiltering* (*SelectPart*) of LOD resources. As in SPARQL, the query can start with a prologue that abbreviates IRIs with namespace declarations [102].

SelectPart (line 2-3) The *SelectPart* part enables subquerying with recommendation results. The surrounding query is a `SELECT` query with a slightly simplified `WHERE` clause (i.e., *RecWhereClause*), with which *postfilter* conditions can be formulated. In this section users have the option to formulate aggregation-based retrieval requests (*Aggregation*).

Aggregation (line 4) This query part handles aggregation-based *postfiltering*. Users specify the IRI (*IRIref*) resource, for which similarity scores of sublevel entities are summarized. Additionally, it is stated how the data should be aggregated (i.e., *SUM*, *MAX* or *AVG*) as well as which variable (*Var*) represents the aggregation-based recommendations in the *postfilter* section. The compiler makes sure that this variable is actually contained in the *RecWhereClause* of the *SelectPart*.

SimProjection (line 5) This part of a SKOSRec query refers to the list of generated suggestions. It defines the recommendation variable to which all similar LOD resources are mapped. In case *pre-* and *postfilter* conditions are set in the other sections; the compiler checks whether the variable (*Var*) appears in these sections as well. Otherwise, the required join operations cannot be executed. It is specified how many recommendations should be displayed (*Integer*) and whether the request will be issued against a default repository or as a *cross-repository request* (*ServiceIntegration*).

ServiceIntegration (line 6) This section indicates that a user intends to receive recommendations from a different SPARQL endpoint than the default endpoint that is stated in the standard configuration of the SKOSRec engine. It specifies the target SPARQL endpoint (*IRIREF*), from which a user wants to receive recommendations. Upon extraction of SKOS annotations from the default source endpoint, a subsequent request is sent to the specified target endpoint.

ItemPart (line 7) This section of a SKOSRec query represents a single preference in the user profile. However, recommendation queries can contain a couple of preference statements. A preference is either expressed as a LOD resource (*IRIREF*) or as a variable, which is bound to a preference query (*VarPart*). In the latter case, the resources are obtained by

matching the graph pattern in the *VarPart* with the triple statements in the repository. The *ItemPart* statement can be additionally supplemented with a concept-to-concept similarity threshold (*Sim*) that triggers concept expansion by means of *flexible similarity detection*.

VarPart (line 8) The construct initiates *preference querying*. Individual likings are expressed as a graph pattern in a *RecWhereClause*. The variable (*Var*) is a placeholder for the LOD resources that will be used to set up the user profile. The compiler has to make sure that the specified variable occurs in the *RecWhereClause* as well. Otherwise, the extraction of LOD resources cannot be carried out correctly.

Sim (line 9) The *Sim* part of a SKOSRec query denotes the threshold of concept-to-concept scores for *flexible similarity detection*. Even though knowledge-based similarity values usually range between 0 and 1, the specification allows a broader range of digits in the *Decimal* datatype to enable application of other similarity metrics, in case they are needed.

Relation (line 10) This section specifies, how concept-to-concept similarity scores should be determined. Users can state whether scores should be „larger“ , „larger than“ or „equal to“ the threshold value.

RecWhereClause (line 11) This clause can occur in three different sections of a SKOSRec query, i.e., either in the *prefilter* (*ItemPart*), the *postfilter* (*SelectPart*) or the *preference filter* section (*VarPart*). The *RecWhereClause* closely resembles the „WhereClause“ of the SPARQL 1.1 specification with all its subsequent parts [102]. Hence, it enables different combinations of graph pattern matching. However, the *RecWhereClause* of the SKOSRec language allows fewer pattern matching expressions than the „WhereClause“ of the regular SPARQL syntax specification [102]. For instance, it is neither permitted to formulate subqueries nor to direct filter requests to more than one repository (see *RecGraphPatternNotTriples* for further explanations). This modification has been made to simplify similarity calculation. After parsing this section, the compiler makes sure that recommendation, *preference* or *postfilter* variables occur at least once in the *RecWhereClause* to guarantee that subsequent steps of the recommendation workflow can be applied on existing LOD resources. In case the *RecWhereClause* comprises a *RecMinusGraphPattern*, it also has to be checked that the variable is not only contained in the *MINUS* part of the group.

RecGroupGraphPattern (line 12) A *RecGroupGraphPattern* can contain one to many basic graph patterns which can be differently combined according to *RecGroupGraphPatternSub*.

RecGroupGraphPatternSub (line 13) This section defines the graph patterns that are applied to identify suitable LOD resources. User filters can be either expressed as basic

graph patterns (*TriplesBlock*) or as combinations of them (*RecGraphPatternNotTriples*). The sequence of different kinds of patterns can be flexibly specified.

RecGraphPatternNotTriples (line 14-15) This query part closely resembles the similar named „GraphPatternNotTriples“ of the SPARQL 1.1 specification [102]. The only difference to SPARQL is that the *RecGraphPatternNotTriples* section is a little more restricted in terms of admissible graph patterns than the „GraphPatternNotTriples“ section of the regular SPARQL specification. For instance, it does not allow to retrieve LOD resources other than from the default graph that is specified in the configuration of the SKOSRec engine. While the ability to perform *cross-repository retrieval* is a central requirement of the system specification, filter patterns have to be applied to datasets that contain SKOS annotations. Hence, they need to be specified before runtime execution to ensure that similarity calculation can be carried out correctly. If a user were able to formulate filter conditions that referred to different RDF graphs and endpoints accordingly, this would not be possible.

RecGroupOrUnionGraphPattern (line 16) This query section marks an alternative graph pattern in the style of the „GroupOrUnionGraphPattern“ of the SPARQL syntax specification [102]. However, as explained before, it neither allows SPARQL-like subqueries, nor federated queries for filtered resources, because this would complicate similarity calculation.

RecOptionalGraphPattern (line 17) In this part of a SKOSRecQuery users are enabled to specify additional graph patterns which might extend the query solution but do not necessarily have to match the data.

RecMinusGraphPattern (line 18) This section handles exclusion of LOD resources for cases when users want to omit certain triple statements from the solution.

6.8.2 Examples

The following paragraphs will present example SKOSRec queries. They illustrate the different ways, in which the language’s syntax elements can be applied to formulate novel types of recommendation requests.

6.8.2.1 On-the-Fly Recommendations

A simple *on-the-fly request* is the most basic form of a SKOSRec query. Listing 6.13 shows the corresponding query to the recommendation task that was introduced in Section 6.4. It retrieves movie recommendations based on the Western movie „They Call Me Trinity“. The request contains a single preference statement, namely the DBpedia resource (`dbp:They_`

Call_Me_Trinity), a prefix with a namespace declaration and a specification regarding the number of suggestions the engine should generate.

Listing 6.13: A simple SKOSRec query for the movie domain (Q1)

```
PREFIX dbr: <http://dbpedia.org/resource/>

RECOMMEND ?movie TOP 5
PREF dbr:They_Call_Me_Trinity
```

In this basic form, the query does not contain any *pre-* or *postfilter* conditions. Thus, the engine simply determines all movies that are similar to the preference in the profile. When executed over the 3.9 DBpedia dataset, the query returns the solution set that is depicted in Table 6.4.

Table 6.4: Result set of Q1

?movie
dbr:God_Forgives...'_I_Don't!
dbr:Ace_High_(1968_film)
dbr:Boot_Hill_(film)
dbr:Trinity_Is_Still_My_Name
dbr:Troublemakers_(1994_film)

6.8.2.2 Recommendations from Flexible Similarity Detection

In contrast to simple *on-the-fly requests*, concept expansion queries rely on the broader semantic space of SKOS vocabularies. For this purpose, concepts that are semantically similar to SKOS annotations of profile items are incorporated into the recommendation model as well. A statement in the request triggers these operations. The DL search scenario of Sect. 6.5 serves as an example for this query type. The request specifies that all SKOS concepts which have a concept-to-concept similarity value higher than 0.75 with the annotations of the input paper will be included in the retrieval procedure (Listing 6.14). As a point of reference, the SKOS-Rec query in Listing 6.15 is only based on direct annotations of the input paper. According to the grammar, simple *on-the-fly requests* do not necessarily need a similarity specification (i.e., *conceptSim* = 1.0), since non-expanded retrieval is the default setting. This statement is optional. Also note that the paper title is not part of the actual SKOSRec query.

Listing 6.14: A SKOSRec query that triggers *flexible similarity detection* in the domain of DL search (Q2)

```
PREFIX econstor: <http://linkeddata.econstor.eu/beta/page/publications/>

RECOMMEND ?paper TOP 3
PREF econstor:21555 (Status Aspirations, Wealth Inequality & Economic Growth) SIM >= 0.75
```

Tables 6.5 and 6.6 list the corresponding solution mappings for each SKOSRec query. The requests were executed over EconStor LOD. From the examples, it can be seen that the spec-

Listing 6.15: A simple SKOSRec query in the domain of DL search (Q3)

```
PREFIX econstor: <http://linkeddata.econstor.eu/beta/page/publications/>

RECOMMEND ?paper TOP 3
PREF econstor:21555 (Status Aspirations, Wealth Inequality & Economic Growth) SIM = 1.0
```

ification of a similarity threshold leads to a re-ranking of results. Section 7.4 will investigate whether this changes user perceptions of recommendation quality.

Table 6.5: Result set of Q2

?paper
econstor:31661 (Consumer heterogeneity evolving from social group dynamics)
econstor:19892 (Risky human capital investment, income distribution and macroeconomic dynamics)
econstor:24983 (Do legal standards affect ethical concerns of consumers?)

Table 6.6: Result set of Q3

?paper
econstor:22657 (Inequality and growth: A joint analysis of demand and supply)
econstor:34662 (Distribution of wealth and interdependent preferences)
econstor:56403 (Wealth concentration over the path of development : Sweden, 1873-2005)

6.8.2.3 Constraint-based Recommendations

The principles of *constraint-based retrieval* have been outlined in Section 6.6. The section also described example requests from different domains, such as multimedia RS or travel destination search. It will now be shown how these requests can be formulated as SKOSRec queries. In addition to separate execution of either *pre-* or *postfilter* conditions, the SKOSRec syntax allows combining them in a single query with a further option to obtain *preference* statements through graph-based retrieval.

Prefiltered recommendations *Prefilter requests* extract suitable LOD resources prior to the process of similarity calculation. Queries with a simple *prefilter* condition retrieve items that are linked to a specific attribute. Example 5 of Section 6.6 can be transferred to the SKOSRec syntax as depicted in Listing 6.16. This request resembles a faceted search query since it combines IR with attribute matching. The user looks for destinations that are similar to his profile and meet the condition of being a place (`rdf:type dbo:Place`) and a nature reserve in Russia (`dct:subject dbc:Biosphere_Reserves_of_Russia`). For comparison, almost the same recommendation query is shown in Listing 6.17 without the respective filter condition.

Listing 6.16: A SKOSRec query with a *prefilter* condition to retrieve Russian nature reserves (Q4)

```

PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbc: <http://dbpedia.org/resource/Category:>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dct: <http://purl.org/dc/terms/subject>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

RECOMMEND ?destination TOP 3
PREF dbr:Lake_Baikal
WHERE
{
  ?destination dct:subject dbc:Biosphere_Reserves_of_Russia .
  ?destination rdf:type dbo:Place .
}

```

Listing 6.17: The SKOSRec query of Listing 6.16 without the *prefilter* condition (Q5)

```

PREFIX dbr: <http://dbpedia.org/resource/>

RECOMMEND ?destination TOP 3
PREF dbr:Lake_Baikal

```

Both queries produce different outputs accordingly (Tabs. 6.7 and 6.8).

Table 6.7: Result set of Q4

?destination
dbr:Golden_Mountains_of_Altai
dbr:Volga_Delta
dbr:Katun_Nature_Reserve

Table 6.8: Result set of Q5

?destination
dbr:Golden_Mountains_of_Altai
dbr:Lake_Oron
dbr:Ogoy_Island

Apart from filter conditions that are formulated as faceted search queries, the SKOSRec syntax facilitates the formulation of far more expressive constraints because of the RDF data model. By these means, known but hidden facts can be better extracted from the data. Consider Example 6 from Section 6.6: Besides the preference for the Lake Baikal region, the user specifies that he would like to receive recommendations for travel destinations in Southeast Asia. However, the DBpedia does not contain any place resources that are annotated with the category `dbc:Southeast_Asia`, even though there exists information on Southeast Asian locations in the dataset. The SKOSRec syntax tackles this quality issue by *expressive* query constraints. In the example, suitable destinations are identified by specifying a graph pattern that matches locations in Southeast Asian countries. Listing 6.18 depicts the corresponding SKOSRec query.

Table 6.9 shows the solution to this query. It contains travel destinations that are both similar to the previously liked destination `dbr:Lake_Baikal` and are located in Southeast Asia. Ecoregions around lakes, rivers or mountains from Southeast Asian countries, such as Cambodia, Vietnam or Malaysia are presented to the user. Hence, by combining similarity calculation with graph-based retrieval, it is possible to identify interesting locations that reflect the informational needs of the user.

Once a retrieval pattern has been found to produce useful results, it can serve as a template for

Listing 6.18: SKOSRec query with an *expressive prefilter* condition to retrieve Southeast Asian destinations (Q6)

```

PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbc: <http://dbpedia.org/resource/Category:>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>

RECOMMEND ?destination TOP 5
PREF dbr:Lake_Baikal
WHERE
{
  ?destination dbo:country ?country .
  ?country dct:subject ?countrySubject .
  ?countrySubject skos:broader* dbc:Southeast_Asia .
  ?destination rdf:type dbo:Place .
}

```

Table 6.9: Result set of Q6

?destination
dbr:Tonle_Sap
dbr:Mekong
dbr:Mount_Kinabalu
dbr:Laguna_de_Bay
dbr:Lake_Toba

other related *constraint-based recommendation* tasks (RQ5).

Postfiltered recommendations In contrast to filtering recommendations before similarity calculation, it is possible to apply user constraints after suitable items have been identified. Therefore, a *postfilter* operation joins the recommendation list with graph patterns. Suggestions are integrated into a SPARQL-like subquery and can thus be modified. By these means, previously identified similar items can be displayed in conjunction with other resources or they might not appear in the final result table at all.

To illustrate the functioning of *postfiltered retrieval*, Section 6.6 has introduced a *rollup request* (Example 8), in which recommendations for city trip destinations were derived from the POIs a user has visited and liked in London. Listing 6.19 shows the corresponding SKOSRec query. The SKOSRec request contains subquery commands that refer to the user's preferred POIs. It specifies, how similarity scores of the generated suggestions should be aggregated (i.e., by summation). The retrieval is based on the 100 POIs that are most similar to the ones in the preference section. Additionally, the subquery part refers to the DBpedia resource that represents London (AGG dbr:London ?city) thus indicating that the list of recommendations should not contain any POIs in this city. The engine displays the three cities, for whom the highest aggregated scores have been detected. In contrast to the *rollup request*, a simple *on-the-fly query* (Listing 6.20) for the same item will in most cases generate entirely different results.

The engine generated two solutions from DBpedia depending on the executed request. While the output table of the *postfilter query* (Tab. 6.10) lists cities that are located near the river

Listing 6.19: SKOSRec query to generate *postfiltered recommendations* (aggregation-based) for a preference profile containing the city of London and related POIs (Q7)

```

PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dul: <http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?city
WHERE
{
  ?poi ?property ?city .
  ?city rdf:type dbo:City .
  ?property rdfs:subPropertyOf dul:hasLocation .
}
LIMIT 3
AGG dbr:London ?city SUM
RECOMMEND ?poi TOP 100
PREF
dbr:Oxford_Street
dbr:Notting_Hill
dbr:South_Bank

```

Listing 6.20: SKOSRec query to generate simple *on-the-fly recommendations* for a preference profile containing the city of London (Q8)

```

PREFIX dbr: <http://dbpedia.org/resource/>

RECOMMEND ?city TOP 3
PREF dbr:London

```

Thames, the solution for the regular SKOSRec query shows other capital cities in the United Kingdom. The disparity between the lists is because other sets of SKOS annotations were fed into the engine. The results indicate that the possibility to choose between different recommendation strategies may be valuable for adopting a query to a specific usage context. The evaluation in Section 7.6 will clarify, whether one method is superior to the other.

Table 6.10: Result set of Q7

?city
dbr:Oxford
dbr:Abingdon-on-Thames
dbr:Wallingford,_Oxford

Table 6.11: Result set of Q8

?city
dbr:Belfast
dbr:Edinburgh
dbr:Cardiff

6.8.2.4 Combinations

The previous paragraphs showed how queries can be formulated with the retrieval approaches of the SKOSRec engine. In some cases, it might be necessary to apply concept expansion to change the ranking of results. In others, it will be better to filter similar items with specific constraints either prior or past to the process of recommendation retrieval. While in the previous examples, these techniques were utilized in separate requests, it is possible to combine them into a single SKOSRec query. The following requests will fuse different retrieval methods

into advanced query forms (RQ6). The next paragraph will showcase scenarios, where the SKOSRec engine only accesses the default LOD repository (i.e., single point of access). After this passage, another paragraph will list two additional recommendation requests as examples for *cross-repository retrieval*.

Single point of access One way to integrate different retrieval approaches is to combine *preference querying* with *constraint-based retrieval*. Consider Example 7 (Section 6.6), that involved the movie director Quentin Tarantino. Since the SKOSRec query language offers the possibility to join query patterns, the engine can better mimic real-world scenarios, where peers ask each other for relevant suggestions. In the SKOSRec query in Listing 6.21 the user states that he liked Quentin Tarantino movies in the past and would like to receive recommendations for directors that shot similar films. Usually, this kind of question can typically only be answered by another human being, e.g., in a personal interaction among friends or in an online forum. The fact that the language facilitates these kinds of queries is one of its key advantages. As Q7, the request represents a *rollup retrieval* pattern that aggregates scores by summation, since a director who created more relevant movies should be ranked higher. Other aggregation strategies could be applied as well. For instance, the average-based aggregation scheme is a feasible alternative. It can account for the fact that directors with many movies may distort the results.

Listing 6.21: SKOSRec query to generate *postfiltered recommendations* based on a *preference query* in the movie domain (Q9)

```
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX yago: <http://dbpedia.org/class/yago/>

SELECT ?director ?movie
WHERE
{
  ?movie dbo:director ?director .
  ?director rdf:type yago:Director110014939 .
}
LIMIT 50
AGG dbr:Quentin_Tarantino ?director SUM
RECOMMEND ?movie TOP 100
PREF [ ?prefMovie
WHERE
{
  { ?prefMovie dbo:director dbr:Quentin_Tarantino . }
  UNION
  { ?prefMovie dbo:producer dbr:Quentin_Tarantino . }
}
]
```

Another way to find out about interesting movie directors is to execute a query based on the characteristics (i.e., SKOS annotations) of a preferred director. Then, the engine retrieves all movies that are connected to similar directors via the property `dbo:director` in the *postfilter* section of the query (see Listing 6.22).

Listing 6.22: SKOSRec query to generate simple *postfiltered recommendations* in the movie domain (Q10)

```

PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX yago: <http://dbpedia.org/class/yago/>

SELECT ?director ?movie
WHERE
{
  ?movie dbo:director ?director .
  ?director rdf:type yago:Director110014939 .
}
LIMIT 50
RECOMMEND ?director TOP 5
PREF dbr:Quentin_Tarantino

```

Table 6.12: Result set of Q9

?director	?movie
dbr:Steven_Spielberg	dbr:Indiana_Jones...
dbr:Steven_Spielberg	dbr:Jurassic_Park_(film)
...	...
dbr:Francis_Ford_Coppola	dbr:The_Godfather_Part_II
dbr:Francis_Ford_Coppola	dbr:The_Godfather_Part_III
...	...
dbr:Christopher_Nolan	dbr:The_Prestige_(film)
dbr:Christopher_Nolan	dbr:The_Dark_Knight
...	...
dbr:Robert_Rodriguez	dbr:Planet_Terror
dbr:Robert_Rodriguez	dbr:Machete_(film)
...	...
dbr:Kevin_Smith	dbr:Clerks
dbr:Kevin_Smith	dbr:Clerks_II

Table 6.13: Result set of Q10

?director	?movie
dbr:Kirk_Douglas	dbr:Scalawag_(film)
dbr:Kirk_Douglas	dbr:Posse_(1975_film)
dbr:Kevin_Costner	dbr:The_Postman_(film)
dbr:Kevin_Costner	dbr:Dances_with_Wolves
...	...
dbr:Gene_Kelly	dbr>Hello,_Dolly!_(film)
dbr:Gene_Kelly	dbr:Invitation_to...
...	...
dbr:Alan_Alda	dbr:Goodbye,_Farewell...
dbr:Alan_Alda	dbr:Margaret'_s_Engagement
...	...
dbr:Steve_Buscemi	dbr:Interview_(2007_film)
dbr:Steve_Buscemi	dbr:Animal_Factory

However, this approach has its weaknesses. For instance, the engine only considers the metadata descriptions of the director, whereas related movies have to be retrieved anew, upon execution of the *postfilter* section. Another weakness is the biased semantics of the query. Usually, when a person favors a movie director, this preference refers to the movies this director has shot and not to the personal features of the person. Two directors might be similar according to certain features such as birth year, nationality or received awards, but the style and genre of the movies they have created can still be different. The queries in Listings 6.21 and 6.22 were executed over DBpedia. Tables 6.12 and 6.13 show the corresponding solution sets.

The ellipses in the displayed tables indicate excluded records. The author omitted them on purpose to improve readability. Both queries returned recommendations lists that are ordered according to the ranking scores of directors, but produced highly different suggestions. While Q9 (Listing 6.21) generated a list of directors who have mostly created action and thriller movies with a twist (i.e., Tarantino's unique filmmaking style [196]), Q10 (Listing 6.22) produces a completely different output. It shows US American actor-directors, whose movies range from musical films (e.g., `dbr:Hello,_Dolly!_(film)`) to western movies (e.g., `dbr:Dances_with_Wolves`) or drama films (e.g., `dbr:Interview_(2007_film)`). Thus, the directors in the solution table of Q10 (Tab. 6.13) seem rather arbitrary and not exceptionally helpful concerning the initial user request. However, the evaluation (Sect. 7.6) will have to verify whether this assumption can be confirmed for a multitude of user requests.

The previous example illustrated, how filtering techniques can be combined to enable novel retrieval strategies that might improve recommendation quality. Another sophisticated retrieval pattern facilitates *cross-domain recommendations*. Listing 6.23 shows an example request that generates movie suggestions based on the preference for a music band (i.e., `dbr:The_Jackson_5`).

Listing 6.23: SKOSRec query to generate *cross-domain recommendations* (Q11)

```
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?movie ?participated ?musicalAct
WHERE
{
  ?movie rdf:type dbo:Film .
  VALUES ?participated { dbo:basedOn dbo:musicComposer dbo:genre dbo:starring }
  { ?musicalAct ?participated ?movie . }
  UNION
  { ?movie ?participated ?musicalAct . }
}
LIMIT 5
AGG dbr:The_Jackson_5 ?movie MAX
RECOMMEND ?musicalAct TOP 100
PREF dbr:The_Jackson_5
WHERE
{
  VALUES ?actType { dbo:MusicalArtist dbo:Band }
  ?musicalAct rdf:type ?actType .
}
}
```

Listing 6.24: SPARQL query to generate *cross-domain recommendations* (Q12)

```
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?movie ?participated ?musicalAct
WHERE
{
  ?movie rdf:type dbo:Film .
  VALUES { dbo:basedOn dbo:musicComposer dbo:genre dbo:starring }
  { dbr:The_Jackson_5 ?participated ?movie . }
  UNION
  { ?movie ?participated dbr:The_Jackson_5 . }
}
LIMIT 5
```

The query contains a *prefilter* condition that triggers retrieval of music artists and bands. Afterward, the scores of the top 100 suggestions are summarized with maximum-based aggregation. Thus, the highest similarity value among the set of music acts determines the ranking score of the film. Also, note because of the aggregation-based retrieval procedure, connections between movies and the LOD resource `dbr:The_Jackson_5` from the profile are excluded from similarity calculation.

The scenario exemplifies how semantic relationships can help to generate suggestions for a target domain (i.e., movies) that are based on a profile containing items from a different source do-

main (i.e., music acts). While regular SPARQL queries perform exact matching of triple statements, the SKOSRec syntax facilitates the integration of similarity- and graph-based retrieval to enrich result lists with potentially relevant items. As a point of reference, consider the SPARQL query for the given *cross-domain* scenario (Listing 6.24). The query omits the similarity calculation part. Thus, it only retrieves movies linked to the resource `dbr:The_Jackson_5`. By this method, the SPARQL results have a better chance of being closely related to the initial user preference. On the other hand, empty result lists can occur when the preferred LOD resource does not have any direct connections to recommendable items. This assumption is backed up by the result lists of the two queries (Tabs. 6.14 and 6.15): The SKOSRec request produces a more comprehensive solution than a regular SPARQL query. Further analyses will have to be conducted to confirm this finding for a sufficient quantity of test cases (see Sect. 7.6).

Table 6.14: Result set of Q11

?movie	?participated	musicalAct
dbr:Moonwalker	dbo:musicComposer	dbr:Michael_Jackson
...	...	
dbr:The_Wiz_(film)	dbo:musicComposer	dbr:Ashford_&_Simpson
dbr:Garfield_Gets_a_Life	dbo:musicComposer	dbr:The_Temptations
dbr:Pipe_Dreams_(1976_films)	dbo:musicComposer	dbr:Gladys_Knight_&_the_Pips
dbr:The_Jacksons:_An_American_Dream	dbo:starring	dbr:Jermanine_Jackson

Table 6.15: Result set of Q12

?movie	?participated	musicalAct
dbr:The_Jacksons:_An_American_Dream	dbo:starring	dbr:The_Jackson_5

Cross-repository recommendations The following two requests will illustrate, how the engine retrieves results from distributed repositories with the help of SKOSRec syntax. The first query refers to Example 9 from Section 6.7. Listing 6.25 shows the corresponding SKOSRec request. The preference section specifies a LOD resource that is contained in EconStor. The engine is only able to process preferences for items from the default LOD repository, i.e., the one that is specified as the standard dataset in the configuration (see Sect. 6.9). The configuration can also list other LOD repositories and their SPARQL endpoints. They are queried when the engine receives *cross-repository requests*. This is the case when a query contains a *ServiceIntegration* statement. The engine looks up the corresponding SPARQL endpoint URL in the configuration and determines the necessary SKOS mappings. Upon extraction of SKOS annotations for the preferred items in the user profile, the system sends a query with mapped concepts to the SPARQL endpoint providing access to the target collection.

The example *cross-repository request* in Listing 6.25 (Q13) was executed over a single SPARQL endpoint, which contained both the source (i.e., EconStor LOD) and the target repository (i.e., AGRIS). It specifies a language constraint (`dc:language = ,eng‘`) to filter relevant items from AGRIS.

Listing 6.25: SKOSRec query to retrieve *cross-repository recommendations* for a single preference statement (Q13)

```

PREFIX econ: <http://linkeddata.econstor.eu/beta/resource/publications/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX dc: <http://purl.org/dc/terms/>

SELECT ?item ?title
WHERE
{
  ?item dc:title ?title .
}
RECOMMEND ?item TOP 5 FROM SERVICE <http://localhost:8890/sparql>
PREF econ:46070
WHERE
{
  ?item dc:language ?lang .
  FILTER (str(?lang) = 'eng')
}

```

Users need to formulate filter conditions for the target repository as *prefilter constraints*. When similar items have been identified, they can only be processed by a *postfilter*, as long as the recommended items are contained in the source collection as well. While it would be desirable to be able to apply user constraints on both the source and the target collections at different stages of the similarity calculation process, this feature is a desideratum that has to be implemented in future versions of the SKOSRec engine. Table 6.16 lists the results for Q13.

Table 6.16: Result set of Q13

?item
agris:GB9708724 (Balance of average annual fresh water inflow into the Adriatic Sea)
agris:XF2002399151 (Training in [...] Future Multilateral Trade Negotiations on Agriculture for Central and Eastern European Countries)
agris:CZ2002001053 (Incidence of bovine tuberculosis in cattle in seven Central European countries during the years 1990-1999)
agris:SE19960095231 (Numerical analysis of vegetation complexes and community diversity of major coastal Dinaric mountains)
agris:GB1997043830 (Immune suppression [...] of commercial broilers in Croatia, Slovenia, and Bosnia and Herzegovina from 1981 to 1991)

In the execution mode of *cross-repository retrieval*, suggestions can also be obtained by a *preference query*. For instance, when a researcher has frequently accessed items from a particular series of publications (e.g., a working paper series by a research institute), the profile can be based on the preference for this collection. Listing 6.26 depicts an example SKOSRec request with a *preference query* part for this test scenario. It represents the information need of an economics researcher who is interested in environmental and agricultural issues in Southeast Europe and has often read papers from the EconStor collection `econColl:263` („Studies on the Agricultural and Food Sector in Central and Eastern Europe, IAMO“). The query triggers profile generation from EconStor. Afterward, the profile encompasses all items that are part of the working paper series. For each one of these items, the SKOSRec engine retrieves annotations from the EconStor repository as well as corresponding SKOS mappings to the AGROVOC thesaurus. Upon retrieval of relevant SKOS concepts, the similarity calculation is executed with

LOD resources from AGRIS. As in the previous example, only English publications are considered. Table 6.17 lists the recommendation results for Q14.

Listing 6.26: SKOSRec query to retrieve *cross-repository recommendations* from a preference profile (Q14)

```
PREFIX econColl: <http://linkeddata.econstor.eu/beta/resource/collections/>
PREFIX dc: <http://purl.org/dc/terms/>

RECOMMEND ?item TOP 5 FROM SERVICE <http://localhost:8890/sparql>
PREF [ ?pref
WHERE
{
  ?pref dc:isPartOf econColl:263
} ]
WHERE
{
  ?item dc:language ?lang .
  FILTER (str(?lang) = 'eng')
}
```

Table 6.17: Result set of Q14

?item
agris:RS2007001152 (Harmful fauna on oilseed rape fields and integral pest management)
agris:XF19960022736 (Proceedings of the International Workshop on Harmonization of Soil Conservation Monitoring Systems)
agris:XF9550373 (Proceedings of the International Workshop on Harmonization of Soil Conservation Monitoring Systems)
agris:CZ2002000513 (International comparison of food consumption and expenses in the Czech Republic, Balkan and other postsocialist countries)
agris:GB9708724 (Balance of average annual fresh water inflow into the Adriatic Sea)

6.9 Implementation

The SKOSRec engine is written in Java. The programming language was chosen because it is platform independent [123] and provides useful software libraries for processing RDF data. For instance, to retrieve LOD resources from SPARQL endpoints, the author applied classes from the Apache Jena framework (version 2.11.2) [14]. Apache Jena is an open source software library that offers ready-to-use tools for the development of Linked Data applications. It provides packages for SPARQL query processing from local or remote LOD repositories. The SPARQL processor for Jena is called ARQ [18].

The libraries supported the development process. Besides of their functionalities, many of the engine's components had to be developed anew to meet the system specification. For instance, the concept of *advanced recommendation requests* required a from-scratch-implementation of certain features. Some system functionalities, such as the SKOSRec compiler, had to be developed as a standalone component. It regulates the correct execution of critical operations of the engine (e.g., graph-based filtering, result set joins, similarity calculation) (Appendix A.4).

An additional feature is the SKOSRec parser. It checks the syntactic correctness of recommendation queries. Since the SKOSRec parser utilizes SPARQL 1.1 syntax elements, a SPARQL grammar definition served as the skeleton for the grammar of the parser. Therefore, the author adapted and enhanced existing components of the SPARQL specification, which can be found in the software package ARQ [230]. The JavaCC (Java Compiler Compiler) library was applied for this purpose. It is a tool that generates Java parsers by conversion of grammar specifications to Java source code [124]. Thus, the SKOSRec parser was semi-automatically developed from a respective syntax specification (Appendix A.2 and A.3).

The engine's similarity calculation unit is almost as equally important as the parser and compiler since it generates suitable suggestions for user preferences (Appendix A.5). Besides the query processing and similarity calculation features, other system components are utility tools. They help to alleviate the retrieval process. Features such as the ranker (ordering recommendations according to similarity scores) (Appendix A.6), the optimizer (implementation of *fast on-the-fly retrieval*) (Appendix A.7) and data processor help to quickly handle LOD resources. Figure 6.12 depicts the overall architecture of the SKOSRec engine with all major and supplementary system features. It can be seen that the engine can interact with both local and remote LOD servers.

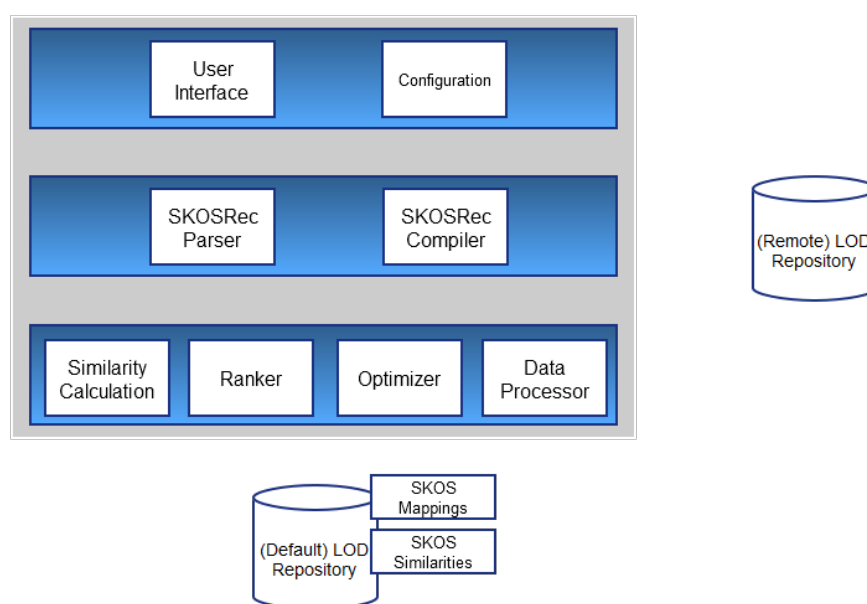


Fig. 6.12: Architecture of the SKOSRec engine

In the standard configuration, the system processes RDF data from a SPARQL endpoint that is accessed over HTTP. For external LOD repositories, this architecture is necessary since it represents the only reliable access method for remote Linked Data collections. Nevertheless, the LOD repository which is queried by default (execution mode of single point access) should

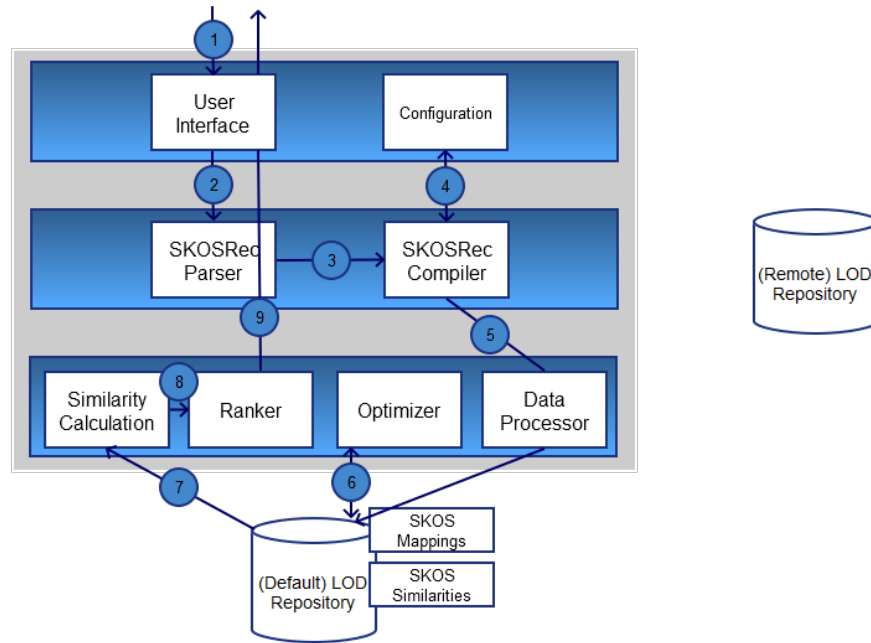
preferably be set up on the same server or local network where the SKOSRec engine is running to speed up response times. However, since the LOD cloud is distributed and decentral by nature, HTTP access of remote SPARQL endpoints is built into the architecture. Aside from the LOD repositories containing item feature data, the engine needs to process SKOS vocabulary information (i.e., SKOS thesauri, SKOS mappings and similarity values of SKOS concept pairs). These data sources either reside in local or external repositories and can be accessed through SPARQL endpoints over HTTP as well. It is expected that, in many cases, SKOS vocabularies reside in the same repository as the respective LOD items.

Each SKOSRec instance loads a configuration object (Appendix A.1). It defines the means of access to the default LOD repository. It is also specified, with which SKOS vocabularies the LOD repository is associated and where the engine can obtain SKOS concept similarities. In case the system processes *cross-repository requests*, the configuration object states additional (potentially remote) LOD repositories and the location of the corresponding SKOS mappings. Thus, the engine can correctly execute similarity calculation for distributed LOD repositories. The architecture does not dictate a particular SPARQL server implementation (e.g., Apache Jena TDB, Fuseki, Joseki, Sesame or OpenLink Virtuoso) since the system is decoupled from the SPARQL endpoint [231]. The prototypical implementation of the SKOSRec engine utilizes the OpenLink Virtuoso server.

Depending on the type of SKOSRec query, the system executes operations in varying sequences. Each of the presented retrieval approaches and their possible combinations can lead to a different workflow. In theory, many workflows could be triggered by a SKOSRec query. However, it is only worth taking a look at the significant differences that can occur in workflow execution. They are caused by the three recommendation approaches of *on-the-fly*, *constraint-based* and *cross-repository retrieval*. Even though these methods can also be combined, their characteristics are best illustrated when describing them as separate workflow instantiations. Figure 6.13 shows the sequence of processing steps for a simple *on-the-fly recommendation* request.

Upon issuing a SKOSRec query, the system parses and compiles it. The engine retrieves recommendations based on the items in the preference part of the query. In case the query contains a *flexible similarity detection* statement; similar SKOS concepts are determined and processed as well (see Tab. 6.18, steps 5. and 7. in italics). If necessary, the optimizer can perform these operations through *fast on-the-fly retrieval* (see Tab. 6.18, step 6. in italics). The engine uses the returned results for item-to-item similarity computation. Afterward, LOD resources are ranked according to their recommendation score. The number of recommendations specified in the query is returned to the user. Table 6.18 outlines the central processing steps of the *on-the-fly recommendation* workflow.

Constraint-based filtering requires some changes in the workflow (Fig. 6.14). For instance, if the SKOSRec query contains all three kinds of filter constraints (a *preference filter*, a *prefilter*, and a *postfilter*), the engine will access the LOD repository more often. Upon parsing, a *preference query* is sent to the SPARQL endpoint determining all items that belong to the user profile.

Fig. 6.13: Workflow of *on-the-fly* retrievalTable 6.18: *On-the-fly* recommendation workflow

Step	Operation
1	Issuing a SKOSRec query
2	Parsing the SKOSRec query
3	Compiling the SKOSRec query
4	Loading information on the LOD repository to be queried from the configuration (default repository)
5	Retrieval and processing of relevant LOD resources and annotations.
6	<i>If the SKOSRec query contains a statement on flexible similarity detection: Determining similar SKOS concepts with regard to the input annotations. Enhancement of the retrieval query with additional SKOS concepts</i>
7	<i>Optimization of the retrieval process (Fast on-the-fly retrieval)</i>
8	Determination of item similarities based on the user profile. <i>If the SKOSRec query contains a statement on flexible similarity detection: Accounting for decreased concept-level similarities</i>
9	Ranking of LOD resources
10	Output of the result set

After retrieval of preferred items, the multiset of potentially relevant LOD resources and their SKOS annotations is determined by querying the SPARQL endpoint a second time. The engine applies the *prefilter* on the triple statements in the repository. Afterward, it identifies similar

LOD resources (i.e., recommendations). The engine sends them to the compiler where they are joined with the *postfilter* statements of the SKOSRec query. The resulting SPARQL query is then issued to the LOD repository to obtain the final solution set.

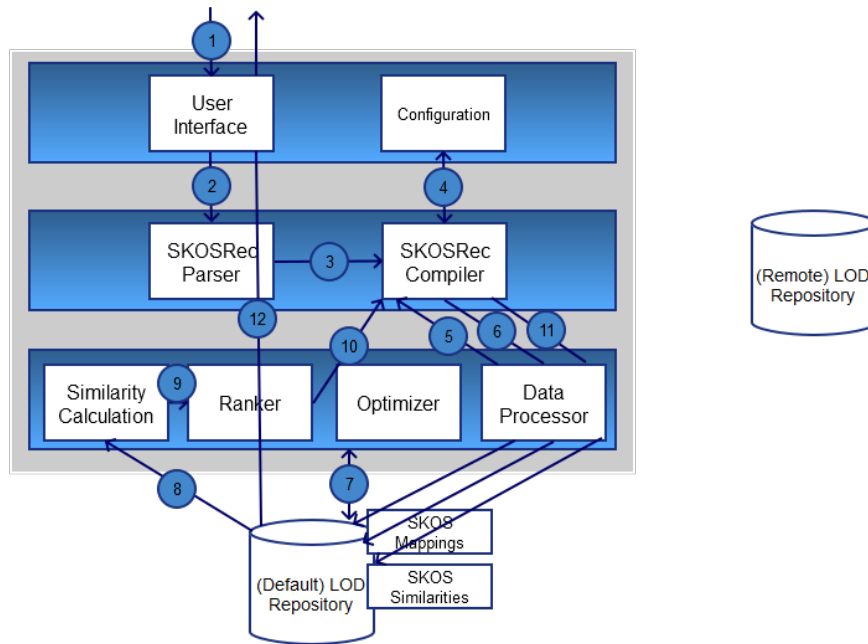


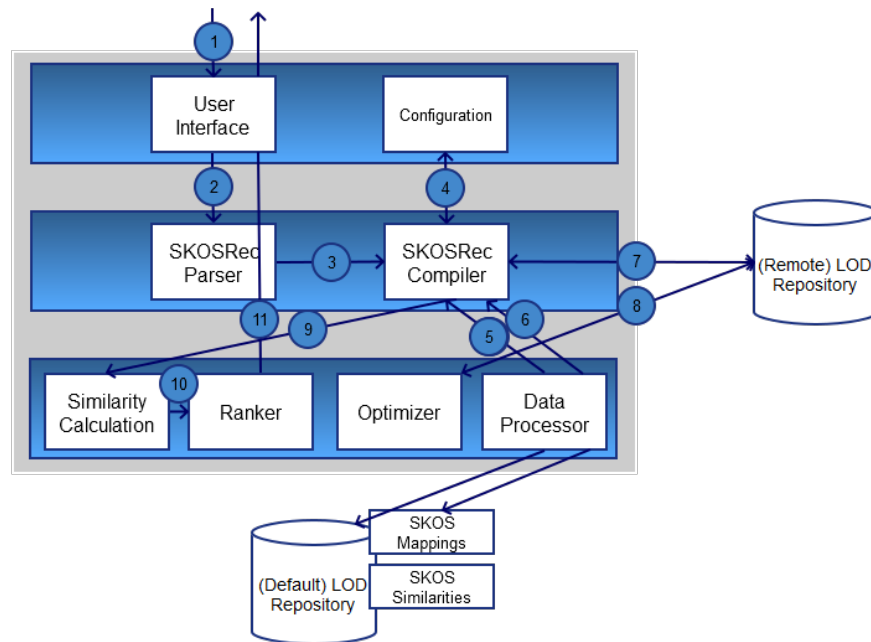
Fig. 6.14: Workflow of *constraint-based retrieval*

Table 6.19 shows the workflow steps of *constraint-based recommendation* retrieval. In case *postfilter* commands are formulated as an aggregation-based request, similarity scores need to be summarized before outputting the final result mapping.

In contrast to the previously presented retrieval approaches, *cross-repository* workflows are characterized by the fact that they acquire information from two different LOD collections. The processor determines SKOS annotations upon receiving the SKOSRec request. The *ServiceIntegration* statement in the original query triggers the execution of a distributed recommendation workflow. Hence, for each of the determined SKOS annotations from the default source repository, the engine looks for mappings to the SKOS vocabulary of the (remote) target LOD repository.

Table 6.19: *Constraint-based recommendation workflow*

1	Issuing a SKOSRec query
2	Parsing the SKOSRec query
3	Compiling the SKOSRec query
4	Loading information on the LOD repository to be queried from the configuration (default repository)
5	Retrieval of preferred items and setting up of the user profile
6	Retrieval and processing of relevant LOD resources upon application of prefilter conditions.
7	<i>Optimization of the retrieval process (Fast on-the-fly retrieval)</i>
8	Determination of item similarities based on the user profile
9	Ranking of LOD resources
10	Sending of recommendation results to the compiler and joining them with postfilter statements
11	Retrieval of the final result set. <i>If the SKOSRec query contains an Aggregation part: Aggregation of similarity scores for sublevel entities.</i>
12	Output to the user

Fig. 6.15: Workflow of *cross-repository retrieval*

The system determines the set of relevant resources and annotations from the target collection when these mappings have been obtained (see Fig. 6.15). The steps that come after this operation are the same as for *on-the-fly recommendations*. Table 6.20 gives an overview of the *cross-repository* workflow.

Now that the technologies and the architecture of the SKOSRec engine have been presented, the following chapter will clarify whether the developed system meets the demands of the requirements specification.

Table 6.20: *Cross-repository recommendation workflow*

1	Issuing a SKOSRec query
2	Parsing the SKOSRec query
3	Compiling the SKOSRec query
4	Loading information on the LOD repositories to be queried from the configuration (default source repository and remote target repository)
5	Retrieval of SKOS annotations from the default source repository
6	Retrieval of SKOS mappings to the SKOS vocabulary used by the dataset in the (remote) target LOD repository
7	Retrieval and processing of relevant LOD resources from the target repository.
8	<i>Optimization of the retrieval process (Fast on-the-fly retrieval)</i>
9	Determination of item similarities based on the user profile
10	Ranking of LOD resources
11	Output of the result set

7 Evaluation of the SKOS Recommender

After having presented the SKOSRecommender, this chapter will clarify whether the recommendation engine meets the demands of the requirements specification. Therefore, suitable evaluation methods have to be identified. Section 7.1 will present common approaches for RS testing and will outline their advantages and disadvantages. The section also introduces the performance dimensions and metrics that will be applied throughout the evaluation. Sections 7.2-7.6 will then describe the specific evaluation methods that have been utilized for testing the SKOSRec engine's performance with regard to different retrieval strategies as well as present the results of these performance tests.

7.1 Evaluation Methods

7.1.1 Prototypical Implementation

The previous section has introduced the SKOSRec engine. The development of the system was based on the requirements detailed in Chapter 4. This process also took into account existing successful approaches to personalized retrieval (Chapter 5). The goal was to adequately address the thesis agenda of harnessing the potential of LOD for RS by applying a systematic approach. Thus, a comprehensive evaluation of the SKOSRec system needs to take into consideration the requirements specification. In detail, the three main categories, namely functional, performance and non-functional specifications have to be evaluated separately. Regarding functional specifications, the SKOSRec engine can serve as a proof-of-concept prototype since extensive examples have demonstrated the successful operation of the separate components in Chapter 6. The SKOSRec engine provides *(fast) on-the-fly recommendations*, *flexible similarity detection* and *cross-repository retrieval* capabilities and a SPARQL-enabled query unit that can process *constraint-based* and *advanced requests*. The detailed technical description of the system, as well as the successfully answered queries, can be regarded as evidence that the prototype fulfills the functional demands [184].

For the two remaining requirement categories (i.e., performance and non-functional specifications) tests have to be carried out differently. For this purpose, suitable evaluation methods need to be identified. The following subsections will present the advantages and downsides of the most common approaches to RS assessment. It will then be described how the engine was evaluated in terms of performance and non-functional specifications with the help of these methods.

7.1.2 Offline Experiments

A comprehensive and statistically sound evaluation of novel recommendation approaches requires access to online platforms with a large user base. Unfortunately, in most cases, researchers are not able to obtain data from a running live application. For this reason, scientists often use historical datasets to simulate recommendation scenarios in offline evaluations [7]. In fact, the majority of research on RS performance has been done offline [133]. Within those experiments, researchers utilize a collection of past user ratings from an existing online platform [7]. Due to the rare number of openly available datasets, extensive research has been conducted with them. Among the most widely used rating collections are the dataset of the Netflix contest and the MovieLens database [107]. In case of utilizing a historical dataset, the focus is on algorithmic performance [133]. Usually, researchers compare two or more recommendation approaches regarding predictive performance [224]. The procedure in these tests is similar to the evaluation of classification tasks, where subsets of user ratings are utilized as training and test data to measure performance. The only difference is that in RS evaluation, separation of data is done entry-wise, instead of row by row. Thus, for each user of the rating matrix, a percentage of the ratings is excluded and a training model is built from the remaining data points [7].

If researchers select training and test samples arbitrarily, evaluation results may be biased. Cross-validation can be applied to avoid these distortions. In this case, user ratings are divided into equal sets that are utilized as a test sample once, while the remaining subsets are applied for training. Evaluation results of the test rounds are finally averaged.

The critical advantage of offline evaluations and the main reason for their popularity is that they represent the most comfortable and most economical form of RS experiments. Historical datasets enable large-scale evaluations on a variety of different test cases. Thus, the reliability of results is increased since the robustness of an algorithm's performance can be tested in various settings [7, 107, 224].

There exist also some downsides to offline evaluation. One is the natural sparsity of RS datasets, which limits the set of items or users that can be tested [107]. Additionally, tests on historical data rely on the assumption that past preferences will resemble future choices [224]. However, user tastes can evolve over time and may thus thwart the *accuracy* of predictions [7]. Another downside concerns the potential bias of historical collections. Since users usually provide feedback for items they liked, researchers should not automatically conclude that the RS accurately predicts preferences for random items, even when performance results are good [5]. The most severe downside of offline evaluations is their limited explanatory power. They can only grasp the performance of an RS for a particular aspect, such as *accuracy* or *diversity*, but fall short, when assessing system performance holistically [107, 224]. Researchers have argued that looking at simple metrics only provides an incomplete approximation of recommendation quality. Instead, when it comes to assessing RS performance, the key defining factor should be *user*

satisfaction [107, 133]. However, *user satisfaction* cannot be measured by a single metric but depends on a variety of factors, which cannot be grasped by an offline evaluation. Hence, to get a comprehensive idea of recommendation quality, evaluations should be complemented with user experiments.

7.1.3 User experiments

User experiments are either carried out in a laboratory setting or on a live platform. In live experiments, a fully deployed commercial system serves as the technical platform. Usually, users are randomly assigned to receive recommendations from two different algorithms [224]. This approach is commonly referred to as A/B testing and resembles experimental setups of clinical trials, where one-half of the subjects gets a gold standard treatment (control group) and the other half experiences a new therapy [7].

Since RS researchers conduct live experiments in a real-world setting, the results provide substantial evidence on the actual quality of recommendations because participants are observed in a natural environment [107, 224]. However, scientists often do not have access to live applications and even if so they might only be able to obtain user data from a single platform. Nevertheless, a comprehensive evaluation requires performance results from diverse usage scenarios and datasets [7]. Additionally, if the algorithm provides low-quality recommendations, the experiment might drive customers away from the live application [224].

An alternative approach is to test the novel retrieval strategies in laboratory experiments, which is the chosen method for the user experiments in the context of the SKOSRec evaluation since the author did not have access to a live application. For laboratory experiments, researchers need to recruit participants actively [132]. The recruited test subjects have to represent the demographic features of real system users as closely as possible to ensure that the experiment provides reliable results. It is also vital to hide the research questions as well as the labeling of the algorithms to avoid biased results. Aside from these requirements, researchers should conduct a pilot study to ensure that the system does not show malfunctions or contains misleading instructions [224].

Laboratory experiments can have a between- or a within-subjects design. In between-subjects designs, users only test a single system (i.e., A/B testing) and researchers compare the evaluation results of the two groups afterward. The between-subjects design more closely resembles a real-world setting since it does not require subjects to interact with more than one engine at a time [224]. Nevertheless, even though participants are randomly assigned to experimental conditions, their differences (e.g., regarding demographics or topic expertise) can potentially have a considerable impact on quality judgments. Hence, when applying a between-subjects design one does not know whether differences in performance are caused by the approaches or the variability among study subjects. Therefore, researchers need to conduct between-subjects experiments with a sufficient number of study subjects to level out these differences.

In within-subject studies, on the other hand, participants test and rate a set of candidate algorithms in a single session. These experiments require fewer participants, while still achieving the same level of statistical power since between-subject variability is not an issue [133]. Moreover, for the evaluation of the customized suggestions of the SKOSRec engine, it is vital to keep the experimental setup as stable as possible in order to only capture differences resulting from different retrieval approaches [29]. That is why the within-subjects design is the primary test method in the user experiments of the SKOSRec evaluation. However, the within-subjects design has some disadvantages as well. When asking participants for their opinions on multiple methods, it is harder to conceal the actual research question. Users might perceive it as unnatural to interact with several engines, which may cause biased evaluations. Additionally, the order in which recommendation lists are presented to users can also have a tremendous effect on the results. For instance, the second time users walk through the evaluation they might be less enthusiastic due to the fact they had already performed a similar task before [133, 224]. Therefore, the result sequence of different algorithms should be chosen randomly to avoid order effects [236].

Besides carefully choosing the experimental design, researchers also have to interpret the user assessments correctly. Even if one algorithm seems to be superior to the other, there is still a possibility that the sample was uniquely suited for a particular approach. Therefore, hypotheses should be formulated before the experiment and significance tests should be conducted afterward to avoid misinterpretations. In this context, it is usually tested whether the Null hypothesis (H_0) that algorithm A does not achieve better results than algorithm B can be rejected. With a significant test, the researcher ensures that observed differences are systematic. Statistical significance is achieved, if the p-value of dismissing H_0 exceeds a certain threshold level. In most cases, the value is set to 0.05 [224].

Experiments may answer a wide range of questions since users are observed while interacting with the system. Additionally, researchers can survey opinions on recommendation quality comprehensively [5, 7, 107]. In contrast to offline simulations, online studies are not limited to the evaluation of one-dimensional metrics but can instead focus on abstract concepts, such as *user satisfaction* [194]. Hence, to determine whether the new approaches of the SKOSRec engine can boost the overall quality of suggestions, user studies should be applied in addition to offline experiments. However, they also have some limitations. For instance, it is often difficult and expensive to recruit large cohorts of participants [5, 224]. Additionally, the recruitment process itself can generate a biased sample since users that agree to participate might have entirely different characteristics than those that do not. Thus, it is likely that recruited study subjects do not form a representative sample of the target population [7, 224]. Therefore, the evaluation of the SKOSRec engine will be conducted as multi-domain tests to level out some of these effects. Aside from the application of suitable methods, appropriate evaluation dimensions have to be defined to ensure that the engine's performance is correctly measured. The following subsection will present the dimensions that play a role in the evaluation of the SKOSRec engine.

Table 7.1: Contingency table for relevance prediction

	Recommended	Not Recommended
Relevant	true positive (tp)	false negative (fn)
Not Relevant	false positive (fp)	true negative (tn)

7.1.4 Performance Dimensions

7.1.4.1 Accuracy

To date, the majority of scientists has focused on the measurement of *accuracy* values to determine RS performance [5, 7, 107, 194, 224]. An *accuracy* metric detects how good the engine predicts true preferences. It is measured by comparing recommendations with real choices. The more the results resemble each other, the likelier it is that the engine provides accurate recommendations. There exists a wide variety of *accuracy* metrics, such as the mean absolute error (MAE), the root mean squared error (RSME) or *precision* and *recall* [107].

Herlocker et al. claim, that for the *find good items* task, which is at the center of this thesis, it is most important to determine the engine's capability to distinguish between relevant and non-relevant items through classification metrics like *precision* and *recall*. They are better suited when users either implicitly or explicitly express their likings through unary ratings (e.g., as preferences for items) as it is done in a SKOSRec query. In the RS literature, classification *accuracy* is often measured in offline evaluations. These experiments determine how many of the recommended items have been marked as relevant by a user in the past (see Tab. 7.1).

In this context, *precision* measures the portion of relevant items among all recommended items (see Eq. 7.1) and *recall* (see Eq. 7.2) indicates the percentage of relevant and recommended items among all relevant items [224]. *Recall* cannot be easily determined in a user experiment, since it is not possible to ask participants to state all relevant items. A possible workaround is to determine the average result list *size* generated by a particular RS algorithm. The *size* can serve as an indicator for actual *recall* values, provided that the majority of suggested items is relevant.

There exists a trade-off between *precision* and *recall* which depends on the length of the recommendation list. For instance, if the number of suggestions is too small, many relevant items may be overlooked by the algorithm (i.e., low *recall*), while if this number is too high, it is more likely that the result list contains too many irrelevant items (i.e., low *precision*). That is why *precision/recall* values are often determined for varying lengths of the recommendation list [7]. However, real-world RS usually fix the number of items to be recommended. Hence, it is reasonable to set the *size* of the result list to medium length (e.g., 10-20 items) in the evaluation of the SKOSRec engine as well [224].

The two measures are often combined in a single metric, called *f-measure* (*f1* score) (Eq. 7.3). It is the harmonic mean of *precision* and *recall* and is used to quantify the overall performance of an algorithm with regard to its classification *accuracy* [107].

$$prec = \frac{\#tp}{\#tp + \#fp} \quad (7.1)$$

$$rec = \frac{\#tp}{\#tp + \#fn} \quad (7.2)$$

$$f1 = \frac{prec \cdot rec}{prec + rec} \quad (7.3)$$

Precision and *recall* are based on a binary relevance model. However, a wide range of score values can reflect the true relevance of a suggestion. For instance, one item may be only slightly relevant, while another one is perfectly in line with the information needs of the user. Hence, to differentiate between nuances of relevance, it is beneficial to let users assess an item's relevance on a scale. Relevance sliders are an established tool in IR evaluations. They capture user judgments on a scale from 0 to 100, where 0 indicates the lowest and 100 the highest relevance [158, 210]. Therefore, an additional *accuracy* metric, called mean relevance score (*mrs*), will be calculated during online experiments to grasp user opinions more comprehensively. *Mrs* is based on slider assessments and measures the average relevance of a recommendation list.

$$mrs = \sum_{i=1}^k \frac{score_i}{k} \quad (7.4)$$

A disadvantage of the previously mentioned metrics is that they do not capture the ranking ability of recommendation approaches. That is why the set of *accuracy* metrics of the evaluation should also contain a ranking measure. Such a metric measures how well the ordering of the suggestions matches the actual user ranking [107]. One option to capture this performance dimension is to compare the engine's predicted item order with the user's choices through correlation analysis. For this purpose, Spearman's ρ or Kendall's τ can be applied [224]. However, these metrics neither take into account that recommendations are usually limited to a certain number of items nor do they model the tendency of users to focus on top-ranked items. Hence, correlation metrics are not particularly well suited for assessments of *top-k recommendation tasks* that are performed by the SKOSRec engine.

That is why utility-based ranking metrics are better applicable for the evaluation in this thesis. These measures assume that higher ranked items are of greater use as they are more likely to be examined by the user. They assign the highest scores to relevant items that are among the top-ranked positions of a result set. Consequently, utility decays, if items are positioned further down the list [7]. The measure *normalized discounted cumulative gain* (*ndcg*) is one of the most

widely applied rank metrics. It rewards accurate item sequences, while only slowly decreasing utility values for low-rank positions through the application of a logarithmic function [7, 224]. *Ndcg* also measures the user gain (*dcg*) (Eq. 7.5). Therefore, the presented ranking (*G*) is set in proportion to the gain a person would receive, when being confronted with a result list in ideal order (*idcg*) [122] (Eq. 7.6).

$$dcg = G[1] + \sum_{j=2}^k \frac{G[j]}{\log_2 j} \quad (7.5)$$

$$ndcg = \frac{dcg}{idcg} \quad (7.6)$$

While accurate recommendations are useful because they build trust in a system [236], they only capture a narrow aspect of performance [164]. Consumers tend to give feedback on prominent items thus perpetuating item popularity even further. Therefore, *accuracy* scores only measure the ability to recommend frequently used items. Suggestions for random items are often not considered. However, researchers and practitioners should also cover this aspect in RS evaluations as the promotion of infrequently used/purchased products (i.e., long tail items) is often as equally important. For instance, an online retailer can tremendously profit from an RS that provides both relevant and unobvious recommendations, since it enables users to explore the product catalog more thoroughly. On the other hand, in the case of an RS only suggesting popular products, it is likely that the user already knows them [7, 107]. A pure *accuracy*-based evaluation disregards these aspects, which is why other performance indicators, such as *novelty*, should be measured as well.

7.1.4.2 Novelty

A novel suggestion is a recommendation for an item the user has not yet rated, used or known about [224]. *Novelty* metrics can be applied in offline as well as online evaluations. In online experiments, researchers directly ask whether a certain recommendation seems familiar. Statements on *novelty* have to be checked against the *accuracy* dimension because users might perceive an item as new but irrelevant [224]. Hence, in concordance with the authors of [236] and [80], *novelty* will be measured as the ratio of relevant documents not familiar to the user (*#nov*) among the total number of relevant documents (*#tp*) (Eq. 7.7).

$$nv = \frac{\#nov}{\#tp} \quad (7.7)$$

In offline experiments, users cannot be directly asked about their opinions. In these tests, the

SKOSRec evaluation has to capture the notion of *novelty* either by determining the popularity of recommended items or by measuring their similarity with the user profile [224]. Castells proposes to calculate the likelihood of the user to choose a certain item i to determine popularity-based *novelty* scores (Eq. 7.8). The likelihood is determined by the ratio of occurrences of a particular item among all user preferences in the database [50]

$$nov_{pop} = \sum_{i=1}^k \frac{-\log_2 p(i)}{k} \quad (7.8)$$

On the other hand, similarity-based *novelty* metrics measure the concordance of recommended items and the user profile [50]. However, the author decided to skip this metric for the evaluation of the SKOSRec engine. The meaningfulness of similarity-based *novelty* scores would have been limited because the engine generates content-based suggestions. Naturally, *novelty* scores would have gone down, whenever the engine identified items that are highly similar to the user profile. However, this is the whole point of using a CB recommendation strategy. Additionally, concordance between the recommendations and the profile might not necessarily indicate that the suggestions are known to the user.

7.1.4.3 Diversity

Whereas in CB RS, the engine generates suggestions that are similar to a user's profile, it can also be beneficial, if result lists are topically diversified. Consider the following example by Ziegler et al.: Suppose a user has stated that he is interested in books written by a certain author. While a recommendation list primarily consisting of books by this author may be perceived as accurate, it is likely that the user is interested in suggestions for other authors. Such homogeneous results can frequently be encountered in CB RS. In Chapter 2 this phenomenon has been described as the *overspecialization problem*. It characterizes the phenomenon that users get easily saturated when being exposed to the same items over and over again [262].

Researchers have proposed to tackle result homogeneity by looking at the recommendation list as a whole since the utility of a result set is more than the sum of the utilities of the individual items. In this line of research, scientists enhanced common RS algorithms with topic diversification features to improve the composition of the recommendation list. They found out that, despite high *accuracy* scores, diversification increases overall *user satisfaction* [134, 262].

However, a one-sided optimization of *diversity* scores will mostly decrease *accuracy* values. Therefore, *diversity* improvements should only cause small *accuracy* losses [260, 262]. In the RS literature, the most widely explored *diversity* metrics are the ones that measure the similarity between each item pair in the recommendation list. The more the items are alike; the less diverse is the result set [224]. The evaluation of the SKOSRec engine will apply the similarity metric (*divC*) by Castells et al. [50]. It determines item-to-item similarity values with the co-

sine similarity metric. Scores are then converted to distance values (Eq. 7.9) and averaged for the entire recommendation list (Eq. 7.10)

$$dist(i_l, i_m) = 1 - \cos(i_l, i_m) \quad (7.9)$$

$$divC = \frac{2}{k(k-1)} \sum_{l < m} dist(i_l, i_m) \quad (7.10)$$

In the online evaluations of the SKOSRec engine users will be additionally asked to assess recommendation list *diversity* with the help of Likert statements (Sect. 7.3). Score values of these assessments will then be summarized in a user-based *diversity* value (*divU*) in order to ensure that the evaluation comprehensively grasps this performance aspect.

7.1.4.4 Perceived Usefulness

Some researchers argue that the aforementioned quality dimensions are not sufficient to evaluate the performance of an RS [107, 132, 134, 193]. Herlocker et al. point out, that a test can only correctly measure RS performance if the particular recommendation task has been defined before the evaluation. For the *find good items* task, *accuracy*, *novelty* and *diversity* scores may not fully reflect the actual quality. It is assumed that users prefer to receive good suggestions instead of recommendations that have been optimized for a particular metric. Therefore, RS should be judged based on their ability to provide useful results. That is why in recent years, there has been a shift towards a more user-centric evaluation of RS [132, 134, 193].

For instance, Knijnenburg et al. have developed a conceptual framework that intends to improve the understanding of the interplay between an RS and its users. The authors claim that system designers can apply this model to evaluate the effects of certain RS features on user opinions, the user experience and user-system interaction patterns. The framework takes into account both personal characteristics of the consumer and situational characteristics [132, 134]. However, other researchers argue that this framework has not been empirically validated [193]. Pu et al. developed another evaluation model, called ResQue. In addition to building upon prior work in the field of RS testing, the researchers utilize ideas from common usability evaluation methodologies, such as the Technology Acceptance Model (TAM) or the Software Usability Measurement Inventory (SUMI). The framework by Pu et al. is a statistically validated model for RS evaluation in online experiments [193]. The researchers apply psychometric constructs to capture user interaction with RS. Pu et al. propose to ask a few questions for each construct. The multi-question approach increases the reliability of survey responses. Pu et al. developed a questionnaire containing Likert items for each construct and formulated hypotheses regarding the causal relations between them. They empirically verified their ideas in large-scale online ex-

periments. Therefore, consumers of various RS platforms assessed the constructs of the framework (Fig. 7.1). The results revealed how the dimensions of algorithmic performance (i.e., *accuracy*, *novelty*, and *diversity*) impact opinions on overall recommendation quality. They showed that *accuracy* is the most critical aspect. However, as argued before, *novelty* and *diversity* have to be taken into account as well [193].

Since the framework by Pu et al. has been empirically verified, it represents a robust methodology to assess the SKOSRec engine. However, the evaluation in this thesis will only focus on a few aspects, because it is primarily concerned with the improvement of RS technology through LOD. Therefore, it is important to measure algorithmic performance (i.e., *accuracy*, *novelty*, and *diversity*) and its impact on the *perceived usefulness* of suggestions. The construct of *perceived usefulness* will be added to the online experiments, because simple quantitative metrics may not fully capture user perceptions. The results by Pu et al. support this assumption, as they found out that algorithmic metrics are considerably correlated with *perceived usefulness* [193]. It is the only construct in the *user beliefs* dimension of the model, that will be considered for the evaluation of the SKOSRec engine because the aspects of *perceived ease of use* and *control and transparency* are related to interface and interaction design.

Additionally, the remaining two dimensions of the model (i.e., *user attitudes*, *behavioral intentions*) (Fig. 7.1), will not be surveyed in the online experiments, as they are only transitively correlated with the *perceived usefulness* of an algorithm. Hence, when the engine provides high-quality recommendations, users will eventually turn to the system, whenever they are in need of a suggestion.

7.2 Evaluation of Performance Requirements

7.2.1 Methods for Performance Evaluation

The previous subsections have presented common metrics and methods for RS evaluation. However, regarding computational performance, the literature does not suggest a standard experimental setting since in many RS, similarity values are calculated offline, which is why runtime performance is not an issue [56, 107, 224]. However, the query-based retrieval approach of the SKOSRec engine requires ad-hoc similarity calculation to enable customized requests. Hence, the evaluation of the system should also include investigations into response times and workloads (RQ9 and RQ10). For this purpose, the approach of *fast on-the-fly retrieval* has been evaluated in all usage scenarios with the help of performance tests. The LOD repositories EconStor and DBpedia that were presented in Section 6.2 were applied for simulation runs. Table 7.2 provides an overview of the number of items and the average amount of SKOS annotations per item in each repository.

While the travel collection is the largest dataset regarding item size, the average number of annotations per item is the smallest in this domain. The other datasets comprise fewer items,

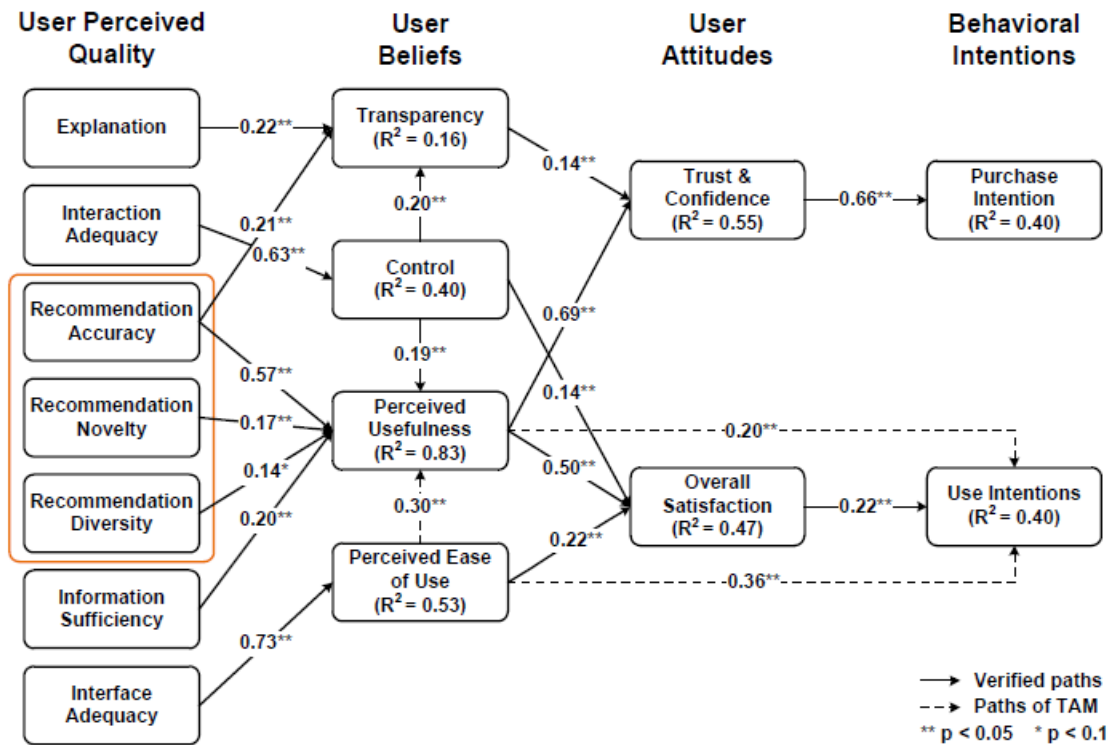


Fig. 7.1: Evaluation framework by Pu et al. [193]

Table 7.2: Overview of the LOD repositories applied in computational simulation runs

Domain	Media			Travel	DL
	Movie	Book	Music		
rdf:type	dbo:Film	dbo:Book	schema:MusicGroup	dbo:Place	swc:Paper
# items	90,063	31,172	86,013	725,546	36,634
(∅) # annotations per item	7.207	4.410	6.060	2.422	6.462

but more annotations than the travel collection. The simulations utilized the local mirrors of DBpedia and EconStor. The tests were set up as follows: The author created user profiles by extracting 100 random LOD resources from each dataset. The SPARQL query in Listing 7.1 was sent to the local database server thereby using the built-in function `RAND()` to randomly select items. A recommendation request was issued to the query processor for each item representing a user profile, upon which the SKOSRec engine generated 20 recommendations. System times were measured for regular and *fast on-the-fly retrieval* on an Intel Core i5 2500, clocked at 3.30 GHz with 8 GB of RAM. The author restarted the database after each simulation run, since the virtuoso server caches query results [223]. Hence, preceding query tables were deleted and execution times could be measured correctly.

Listing 7.1: SPARQL query to create user profiles

```

1 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2
3 SELECT DISTINCT ?item
4 WHERE {
5   ?item rdf:type ?type .
6   ?item <ANNOT.PROP> ?subject .
7 }
8 ORDER BY RAND()
9 LIMIT 100

```

7.2.2 Results of the Performance Evaluation

Response times and workloads (i.e., number of records to be processed) were measured in the regular (i.e., *on-the-fly retrieval*) and the optimized execution mode (i.e., *fast on-the-fly retrieval*) of the SKOSRec engine. Table 7.3 shows that the proposed optimization approach considerably diminished workloads, as is indicated by the reduced number of triple statements in each (sub-)domain.

Table 7.3: Results of the performance test (workloads)

Dataset	t-test	(\emptyset) # records (regular)	(\emptyset) # records (optimized)	(\emptyset) reduction ratio
DL	$t(8.372) = 99, p < 0.001$	3812	1459	44.8%
Travel	$t(3.830) = 99, p < 0.001$	621	218	36.0 %
Movie	$t(5.828) = 88, p < 0.001$	23542	20305	42.6%
Music	$t(7.202) = 99, p < 0.001$	7662	4145	52.1%
Book	$t(8.956) = 99, p < 0.001$	1206	303	51.9 %

The mean reduction ratio was determined by averaging single reduction scores (Eq. 7.11) for each recommendation request.

$$reduction\ ratio = \left(\frac{regular - optimized}{regular} \right) \cdot 100 \quad (7.11)$$

Additional t-tests on the differences in means of the two methods confirmed that the fast retrieval approach significantly minimizes data transmission. Figures 7.3-7.6 also show that the optimized approach helps to avoid exponential growth of processing periods (RQ10).

Moreover, the experiment investigated whether the system running in optimized mode provides quick answers to single item requests (RQ9). Table 7.4 reveals that the fast retrieval method, on average, generated recommendations in under a second in 4 out of 5 domains. The system took longer only in the movie domain. Here, responses were mostly provided in under 2 seconds, which is still within reasonable limits. The simulation runs also showed that *fast on-the-fly retrieval* reduced processing times compared to regular retrieval. Subsequent t-tests were carried out to examine whether the reduction of mean scores could be statistically confirmed.

Table 7.4: Results of the performance test (execution times)

Dataset	t-test	(\emptyset) exec. time in ms (regular)	(\emptyset) exec. time in ms (optimized)	(\emptyset) reduction ratio
Dig. Library	$t(1.264) = 99, p = 0.105$	630	196	-1.5%
Travel	$t(0.899) = 99, p = 0.186$	134	73	-16.5%
Movie	$t(4.921) = 88, p < 0.001$	9961	1910	65.2%
Music	$t(2.146) = 99, p < 0.05$	1022	501	4.4%
Book	$t(1.879) = 99, p < 0.05$	834	92	23.2%

The test verified the hypothesis for the majority of domains (i.e., the multimedia domains). However, it was not confirmed for the travel and DL datasets. Here, even though mean processing times were considerably smaller when applying the optimized execution mode, the statistical analysis found no significant differences. The corresponding runtime diagrams (Figs. 7.3-7.6) explain these results further. In these domains, there were only a few test cases for which the fast retrieval approach tremendously reduced processing times. While the remaining simulation runs led to reduced workloads, the optimized approach induced some overhead that decreased computational performance. The reduced number of records in the respective data collections may have caused this outcome. In DBpedia, location-specific LOD resources, on average, only have a few SKOS annotations, while EconStor is a small dataset, from which the engine can extract records more quickly.

In summary, it can be concluded that *fast on-the-fly retrieval* improves scalability, especially, if large workloads are involved (RQ10). However, in domains, in which only a small amount of data is processed, the regular approach already provides fairly quick responses (RQ9). Additionally, *fast on-the-fly retrieval* can help to reduce processing times, when datasets contain many items and SKOS annotations.

7.3 Evaluation of General Recommendation Quality

7.3.1 Methods for the Evaluation of General Recommendation Quality

Apart from performance simulation runs, quality tests are equally necessary for the assessment of the SKOSRec engine. As has been outlined before, user experiments are a well-suited method for RS validation. In the context of the SKOSRec evaluation, they were carried out as web-based studies since the author did not have access to a real-world application. The web setting helped to reach out to more participants with diverse demographic backgrounds, which would not have been possible in a laboratory.

The experiments were conducted for the defined usage scenarios with the corresponding LOD

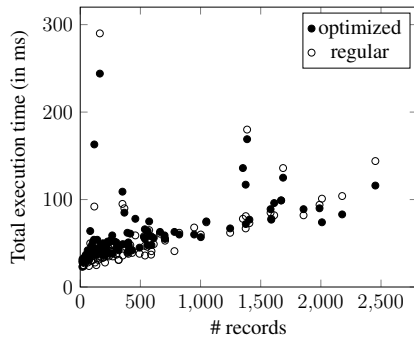


Fig. 7.2: Exec. time in the travel domain

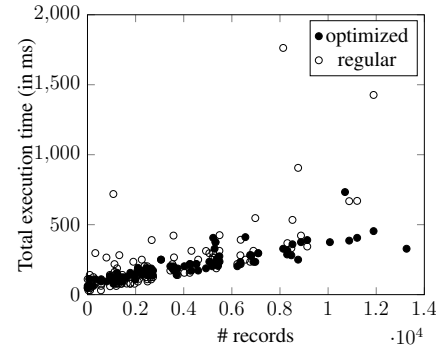


Fig. 7.3: Exec. time in the DL domain

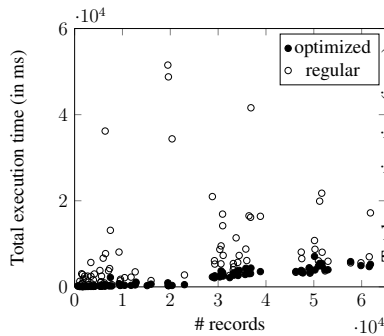


Fig. 7.4: Exec. time in the movie domain

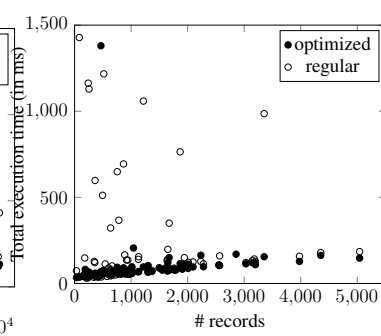


Fig. 7.5: Exec. time in the book domain

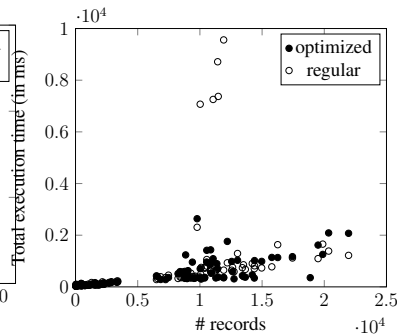


Fig. 7.6: Exec. time in the music domain

repositories. The database containing the local mirrors of DBpedia and EconStor LOD ran on a virtual machine. The machine had one CPU core and 14 GB of RAM allocated to it. Besides the triple store, a web application was developed and hosted on the server. The website ran on an Apache Tomcat 7 application server and was implemented with the help of Java servlet technology [16, 125].

The author conducted the online experiments between April 2016 and January 2017. The study series started with an experiment on LOD-enabled retrieval in the domain of DL search. Upon setting up the web application, a pretest was carried out to check for potential pitfalls, such as misleading navigation. Two users ran step by step through the web interface. Based on their feedback, the author slightly modified the interface (e.g., through rephrasing user instructions) to increase the chances of successful completion.

Afterward, the recruitment process started. The author posted a link to the study alongside some additional information in two online forums for economics in higher education. Additionally, the experimenters (i.e., the author and an assistant) contacted economics researchers by phone and e-mail. For this purpose, a list of major German institutions in this field, e.g., economics chairs at universities, was compiled. The experimenters worked down the list and contacted affiliated researchers. Participation was incentivized with five 20.00€ Amazon gift vouchers. Study subjects were entered into a prize draw upon successful completion of the survey. The experiment took place between April and June 2016. In total, 39 subjects with an economics

background participated.

The remaining studies (i.e., the experiments on multimedia and travel RS) were carried out differently. While for the usage scenario of DL search, participants had to be directly contacted, due to their specified field of interest, this was not necessary for the other domains. It was assumed that the Wikipedia-based experiments would appeal to a broader audience and that most participants of a randomly selected sample would be able to generate an individual profile containing LOD resources from DBpedia. The author recruited subjects through clickworker.com [54]. The crowd working platform advertised the experiments. Participants received a compensation fee (i.e., between 1.00€ and 1.50€ for a completed survey). 103 participants took part in the study on travel destination search. In the multimedia domains, 154 subjects evaluated the SKOSRec engine.

The travel experiment took place in August 2016. The study on movie RS was conducted in October 2016, and the two remaining experiments on music and book RS were both carried out in January 2017. Table 7.5 shows the organizational details of the study series.

Table 7.5: Details of the study series

Experiment	Time of Conduction	Participants
Digital Library	April - June 2016	39
Travel	26.08.16-01.08.16	103
Movie	22.10.16	50
Music	11.01.17	53
Book	13.01.17	51

The first part of the experiment collected data on participants' demographics and their consumption and search habits (see Fig. 7.7). This was primarily done in order to prepare subjects for the test cases to be completed during the study.

After the experiment in the domain of DL search had already been carried out, the author added a page to the survey template of the first section. It informed participants about the context and the procedure of the study. The web form also assured subjects that no deception was involved in survey completion and underlined the participants' right to refrain from the experiment at any time. Additionally, participants were asked to confirm that they were 18 years or older. The web form obtained the subjects' direct consent to take part in the experiment under the provided conditions thereby helping to carry out the study in an ethically sound manner [244]. The majority of the form's content was taken from a template consent form [211]. The author adapted it slightly to the different usage scenarios. Figure 7.8 shows the consent form of the experiment on music RS. In the sequence of survey pages, the template was placed before the demographic section and served as a gateway for participation.

In the DL study, the experimenters provided context information either over the phone or via e-mail. For the DL experiment, it is assumed that, through their participation, subjects implicitly gave their consent. They were comprehensively informed about the study during the recruitment

Recommender Systems
User Studies

[Home](#) [Contact](#)

Media Recommender Systems - Music

Please start by answering a few general questions about yourself and your music search habits.

Demographic Information

Q1. Your age range
under 20 ▾

Q2. Your Gender
female ▾

Music

Q3. What is your favorite genre?
Alternative Music ▾

Q4. How many hours do you spend listening to music per day (on average)?
1-2 hours ▾

Music Search

Q5. How do you hear/find out about interesting music acts?
(Multiple responses are possible)

Family and friends
 Media coverage (TV, magazines, newspapers)
 Media store
 Local library
 Online communities (e.g. Last.fm, Slacker)
 Onlineshops, Streaming websites (e.g. Amazon, Spotify)
 Others

Q6. Please complete the following sentence: Finding interesting music acts is ...
 very easy easy manageable hard very hard

Fig. 7.7: Music - Demographics section (page 2)

process. In case they had any reservations they could have withdrawn from it right away.

After the preparation phase, participants started with the first test case (TC1) in each usage scenario. In this test case, the performance of the SKOSRec engine was assessed in the simple execution mode of *on-the-fly retrieval*. It was done to gather data on the usefulness of suggestions resulting from a baseline CB recommendation method, with which more advanced strategies could be compared at a later stage. The author investigated, how *accuracy*, *novelty* and *diversity* affect *perceived usefulness*. In TC1, users stated their favorite items through a web form. In the DL experiment, participants only had to enter one publication that reflected their research interest. This approach was taken because, even though EconStor is among the largest Open Access servers in its field, it is still merely a repository server for grey literature [78]. Consequently, it could not be presumed that participants would find numerous research papers on their specific topics of interest easily. Thus, the user profile was limited to one item to prevent

Recommender Systems
User Studies

[Home](#) [Contact](#)

Media Recommender Systems - Music

Thank you for participating in our experiment on media recommender systems! Your help is very much appreciated. The study examines the usefulness of recommendation algorithms when searching for music acts. Before taking part in this experiment, please read the following consent form and click on the "I Agree" button at the bottom of the page if you understand the statements and freely consent to participate in the study.

Consent Form

This web-based experiment is designed to understand preferences for different recommender systems' algorithms in the music domain. The study is being conducted by Lisa Wenige (M.Sc.), research assistant at the Chair of Business Information Systems, Friedrich Schiller University Jena, Germany. No deception is involved, and the study involves no more than minimal risk to participants (i.e. the level of risk encountered in daily life).

Participation in the study typically takes 20 minutes and is strictly anonymous. Participants start by providing demographic information and answering a few questions regarding their habits when searching for music acts. Then, they will be asked to evaluate recommendation results in 4 different music search test cases. Sufficient instructions are provided for each test case.

All responses are treated as confidential, and in no case will responses from individual participants be identified. Rather, all data will be published in aggregate form only. Participants should be aware, however, that the experiment is not being run from a "secure" https server of the kind typically used to handle credit card transactions, so there is a small possibility that responses could be viewed by unauthorized third parties (e.g. computer hackers).

Many individuals from previously conducted similar experiments found it to be at least interesting, if not enjoyable, and no adverse reactions have been reported thus far. Participants from clickworker.com will receive a monetary compensation. Other visitors to this web site are welcome to complete the study, although they will receive no compensation. Participation is entirely voluntary. Participants are free to refuse to answer any question or withdraw from the experiment at any time. The decision about whether to take part in the study will not affect any future interactions of the participants with media content providers or their relations with the Friedrich Schiller University Jena.

Participants who have further questions about this study or their rights, or wish to issue a complaint or concern may [contact us](#) via phone or e-mail.

If you are 18 years of age or older, understand the statements above, and freely consent to participate in the study, click on the "I Agree" button to begin the experiment.

Fig. 7.8: Music - Consent form (page 1)

that potential study subjects withdrew from the experiment. For the DBpedia experiments, the author assumed that participants would find more than one item that reflected their preferences. Thus, subjects had to specify three of their favorite items in these studies (Fig. 7.10). However, since withdrawal from DBpedia experiments should be also prevented, error messages were not thrown, even if participants provided fewer objects than required. Hence, despite the three-item input form of the website, participants could carry on with only 1 or 2 favorite items in their profile. The initial statements regarding user tastes represented a vital step, because personalized recommendations were based on these items. Thus, in cases, where users did not state any items, the application showed an error page to the user (Fig. 7.9).

The process of profile generation was facilitated by a specifically tailored interface (see Fig. 7.10), where subjects could search for items through an autocomplete field that applied the technology of Asynchronous JavaScript and XML (AJAX) [126].

It enabled users to type in keywords (e.g., the name of a music act or the author of a book), for which matchings were instantaneously displayed without reloading the site. Hence, participants could quickly find and select their items of interest. Whenever a user entered a query through the AJAX interface, the system matched the query keywords against an Apache Solr/Lucene search index [15]. The index was built with item metadata, which had been extracted from the

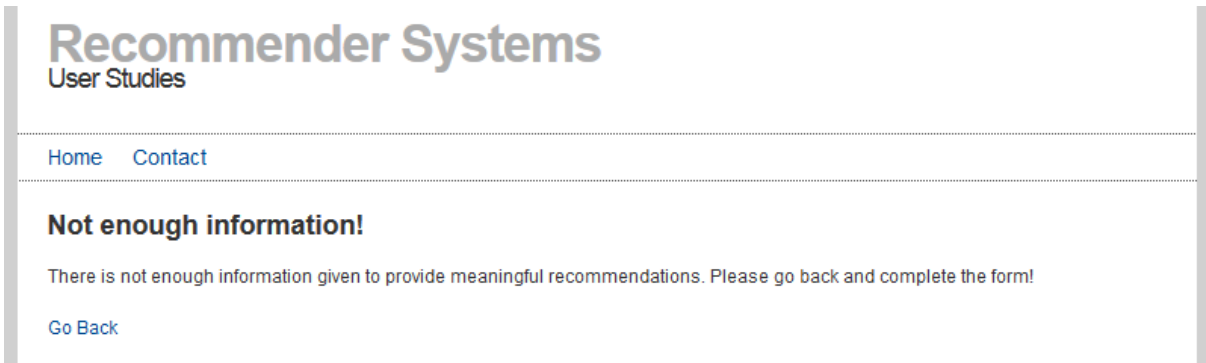


Fig. 7.9: Error page

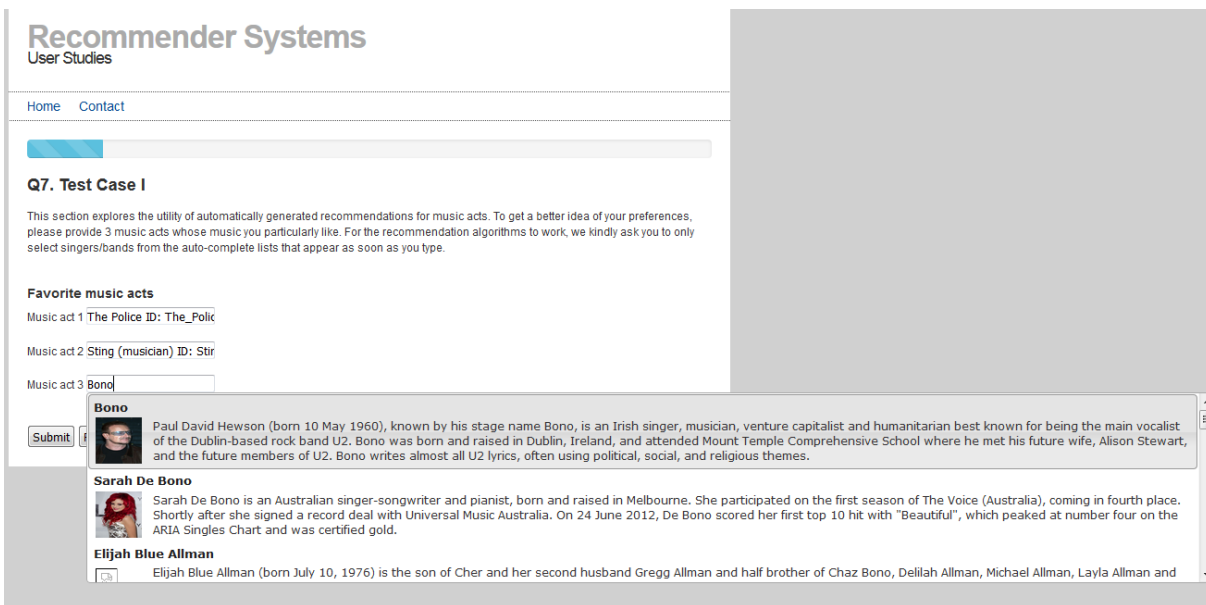


Fig. 7.10: Music - User profile generation, TC1 (page 2)

respective LOD repositories prior to setting up the website. For the web experiment in the usage scenario of DL search, the index comprised the author, title and abstract of economics research papers, while the index for DBpedia resources was set up with the names and abstracts of items. Whenever available, information on thumbnail pictures were saved in the index to be ready for display in the AJAX interface. In case the Apache Solr/Lucene search engine found index keywords that matched the user query, metadata information for these items were immediately shown on the screen. This information was displayed to help participants make an informed decision on whether the result list contained suitable items for profile generation.

Aside from the AJAX feature, the appeal of the interface was increased by a progress bar at the top of the webpage (see Fig. 7.10). In conventional paper-based studies, subjects usually see their current advancement level, while in a web survey they cannot check their progress immediately. This is especially true for studies that apply a screen-by-screen navigation, which was the chosen approach for the web experiments of this thesis. Progress bars can be applied to handle this problem. They prevent users from tiring of the questionnaire and withdrawing from

the study altogether by showing, how close participants are to complete the experiment [70]. Upon profile generation, the SKOSRec engine calculated *on-the-fly recommendations*, which were shown to users in a separate screen. Suggestions were displayed with a sufficient amount of information e.g., thumbnails, labels or a short description to help participants assess the quality of the recommendation. In addition to this information, users could also access the respective Wikipedia article or the EconStor URL by following the hyperlink on the label. Item-level assessments of recommendation results were partly carried out with the help of relevance sliders. They let participants state a degree of relevance for each item, instead of asking them to make a binary decision on item relevance. Scores were handled as points on a 0 to 100 scale of the relevance slider (outer left side: lowest relevance, outer right side: highest relevance, see Fig. 7.11). This approach was taken because it conveniently enabled calculation of the *accuracy* metrics presented in Subsection 7.1.4.

In addition to relevance sliders, the evaluation interface also contained a section, where subjects had to tick a radio button that indicated, whether an item was new. In concordance with the *novelty* definition from Subsection 7.1.4, novel items were assumed to have not yet been consumed by the user or were not familiar or both. No distinctions were made between these subnotions of *novelty* to avoid confusion as well as to keep the number of questions as low as possible. This trade-off between comprehensively capturing users' opinions on recommendation quality and the maintenance of a clear questionnaire had to be carefully balanced out during the study series.

Once participants had stated their item-level assessments with regard to *accuracy* and *novelty*, they were asked to judge the quality of the recommendation list as a whole. In this section, participants stated their agreement for several Likert questions measuring the concept of *perceived usefulness*. When the first two experiments of the study series (i.e., the web experiments on DL search and travel RS) were carried out, only two Likert questions were posed, in order to keep the required user effort at a minimum level. The following list shows the questions that were posed in the experiment on DL search.

- The recommendations [...] better fit my research interests than what I may receive from a research fellow.
- I feel supported to find relevant publications with the help of the recommendations.


The questions were only slightly adapted to reflect the specificities of the travel domain. After the first two experiments had been carried out, Cronbach's α values for the two Likert items reached only acceptable levels (see Subsect. 7.3.2), which is why for subsequent web experiments further questions were added to this section to increase the reliability of the scale. The questions were formulated in reference to the Likert items by Pu et al., which are assumed to measure the concept of *perceived usefulness*. They were adapted to each of the three remaining multimedia usage scenarios accordingly [193]. Figure 7.12 shows an example collection of Likert questions for the domain of music RS.


Q7. Results of Test Case I


Below, you find recommendations that are provided to you according to your favorite music acts in test case I. For each music act in the recommendation list, move the slider to evaluate its relevance (outer left side: lowest relevance, outer right side: highest relevance). In addition to that, tick one of the buttons in the outer right column to indicate, whether a music act is new to you or not. Please examine the overall usefulness of the recommendation list on the bottom of the page.

Please open hyperlinks in a new tab. Otherwise your evaluations are gone.

Your favorite music acts were...

 [The Police](#)

 [Sting \(musician\)](#)

 [Bono](#)

Recommendations




Title	Relevant	New
 <p>Paul McCartney Sir James Paul McCartney, MBE (born 18 June 1942) is an English musician, singer-songwriter, multi-instrumentalist and composer. With John Lennon, George Harrison and Ringo Starr, he gained worldwide fame as a member of the Beatles, and his collaboration with Lennon is one of the most celebrated songwriting partnerships of the 20th century. After the band's break-up, he pursued a solo career, later forming Wings with his first wife, Linda, and singer-songwriter Denny Laine.</p>		<input type="radio"/> yes <input type="radio"/> no
 <p>David Bowie David Robert Jones (born 8 January 1947), known by his stage name David Bowie, is an English musician, singer-songwriter, record producer, actor, and arranger. Bowie has been a major figure in the world of popular music for over four decades, and is renowned as an innovator, particularly for his work in the 1970s. He is known for his distinctive voice as well as the intellectual depth and eclecticism of his work.</p>		<input type="radio"/> yes <input type="radio"/> no
 <p>John Lennon John Ono Lennon, MBE (born John Winston Lennon; 9 October 1940 – 8 December 1980) was an English musician, singer and songwriter who rose to worldwide fame as a founder member of the Beatles, one of the most commercially successful and critically acclaimed acts in the history of popular music. With Paul McCartney, he formed one of the most celebrated songwriting partnerships of the 20th century.</p>		<input type="radio"/> yes <input type="radio"/> no




Fig. 7.11: Music - Results part I, TC1 (page 4)

In addition to Likert statements on *perceived usefulness*, the author included two items that captured the notion of recommendation list *diversity* ($divU$). Hence, even though in the backend of the web application, *diversity* was determined through calculation of content-based dissimilarity scores ($divC$), user assessments on *diversity* ($divU$) were additionally measured with the following Likert items:

- The recommendations of the list are diverse.
- The recommendations of the list are similar to each other.

Analogously to the Likert items for *perceived usefulness*, the above mentioned statements were taken from the RS evaluation framework by Pu et al. The scoring is reversed in the second item to increase the reliability of the scale [193].

After participants had provided the required information, their answers were saved in a log file

 Robin Gibb Robin Hugh Gibb, CBE (22 December 1949 – 20 May 2012) was a singer and songwriter, best known as a member of the Bee Gees, co-founded with his fraternal twin brother Maurice and older brother Barry. Their younger brother Andy was also a singer. Born in the Isle of Man to English parents, the family later moved to Manchester before settling in Redcliffe, a suburb of Brisbane, Australia.	<input type="range"/>	<input type="radio"/> yes <input type="radio"/> no
 Elton John Sir Elton Hercules John, CBE (born Reginald Kenneth Dwight on 25 March 1947), is an English rock singer-songwriter, composer, pianist and occasional actor. He has worked with lyricist Bernie Taupin as his songwriter partner since 1967; they have collaborated on more than 30 albums to date. In his four-decade career John has sold more than 250 million records, making him one of the most successful artists of all time.	<input type="range"/>	<input type="radio"/> yes <input type="radio"/> no
 Robert Plant Robert Anthony Plant CBE (born 20 August 1948) is an English musician, singer and songwriter. Best known as the lead vocalist and lyricist of the rock band Led Zeppelin, he also had a successful solo career. With a career spanning more than 40 years, Plant is regarded as one of the most significant singers in the history of rock music, and has influenced contemporaries and later singers such as Freddie Mercury, Axl Rose and Chris Cornell.	<input type="range"/>	<input type="radio"/> yes <input type="radio"/> no

Statement	Agreement
The recommendations better fit my preferences than the suggestions I would receive from other sources (see Q5).	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
I feel supported to find interesting music acts with the help of the recommender.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The recommended items influence my selection of music acts.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The recommended items effectively helped me to find what I like.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
I feel supported in selecting the items to buy with the help of the recommender.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree

Fig. 7.12: Music - Results part II, TC1 (page 4)

and they were navigated to subsequent test cases (i.e., Tflex, TC2-TC4), where they worked on other more advanced retrieval tasks. Table 7.6 provides an overview of all test cases of the experimental series. Sections 7.4-7.6 will elaborate on the test settings for these kinds of recommendation requests.

Table 7.6: Test cases in the web experiments

Test Case	Evaluation Purpose	DL	Travel	Movie	Music	Book
TC1	On-the-fly Recommendations (Baseline)	✓	✓	✓	✓	✓
Tflex	Flexible Similarity Detection	✓	✗	✗	✗	✗
TC2	Constraint-based Recommendations	✓	✓	✓	✓	✓
TC3	Advanced Recommendation Queries (Roll-up)	✗	✓	✓	✓	✓
TC4	Advanced Recommendation Queries (Cross-Domain)	✗	✗	✓	✓	✓

7.3.2 Results of the Evaluation of General Recommendation Quality

In total, 296 persons participated in the web experiments. The demographics were as follows: The sample contained answers from both genders with a slight overrepresentation of male participants (53.7%). Male subjects were the majority in the studies on DL search, movie and book RS (Fig 7.13). The prevailing number of participants was between 20 and 40 years of age. The domain-wise age distribution is depicted in Figure 7.14. Participants younger than 20 or older than 60 years formed only a tiny proportion of the sample.

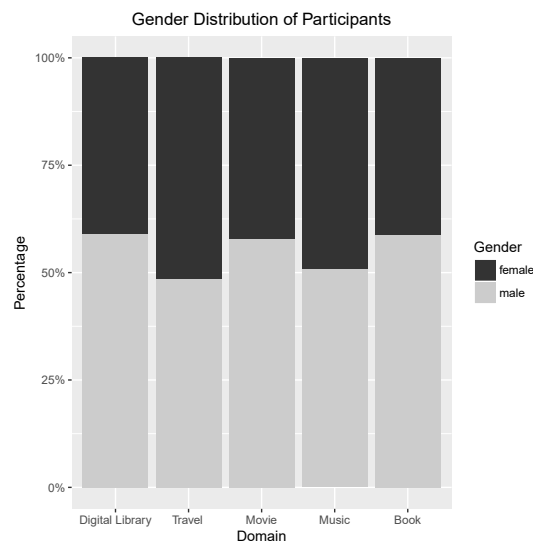


Fig. 7.13: Domain-wise gender distribution

Aside from the small percentage of young study subjects (< 20 years), which can be explained by the age barrier for participation, the samples can be regarded as representative of regular Internet users both in terms of gender and age.¹

Besides the demographic questions, in each usage scenario the questionnaire contained a section on the consumption behavior of participants. This section attuned subjects to the experimental setting. It also gathered answers to the statement „Finding interesting items in my domain of interest/expertise is...: 1 („very easy“), 2 („easy“), 3 („manageable“), 4 („hard“), 5 („very hard“). Response rates for the corresponding categories are depicted in Figure 7.15. The diagram shows that the vast majority of participants perceives the search as at least „manageable“ of which many even call it „easy“ or „very easy“. An interesting finding is that subjects, who took part in the DL experiment regard the search as more easy as in the other domains. This is surprising, since retrieval tasks in the context of scientific research are mostly carried out as a vocational endeavor, while in the remaining domains, a search is most often conducted to organize leisure activities.

¹Cf. the following webpage for an overview on gender and age distribution of Internet users: [116].

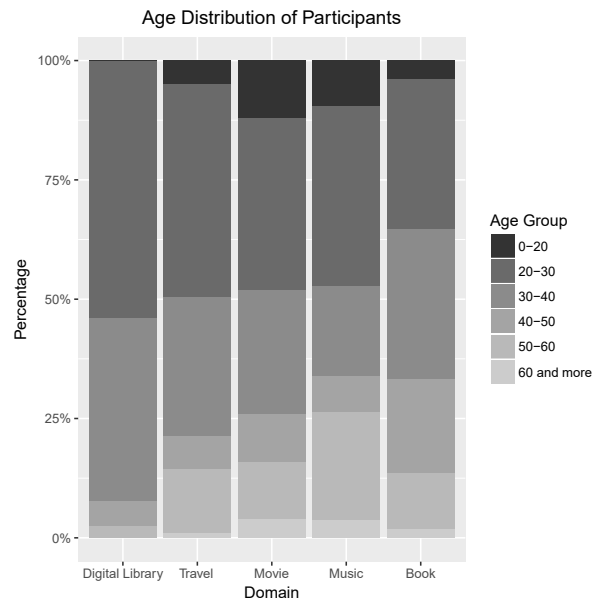


Fig. 7.14: Domain-wise age distribution

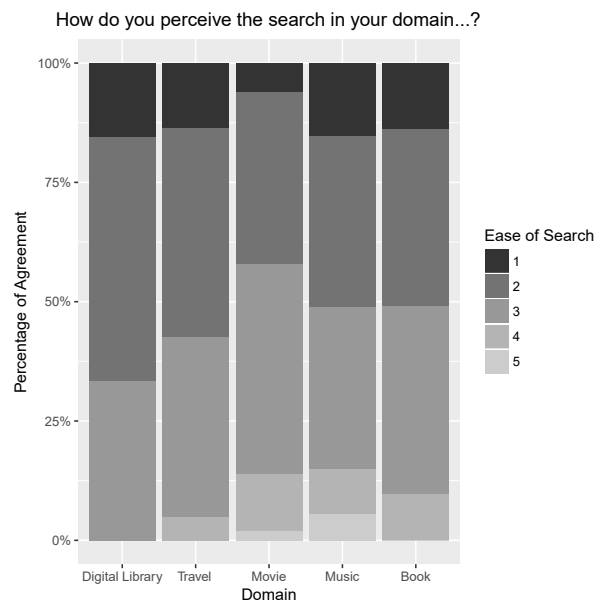


Fig. 7.15: User perceptions of the search in their domains of interest

Throughout the experiments, participants completed between 3 to 4 retrieval tasks during a single session. This amount of test cases was considered as the maximum that subjects could be asked to work on in relation to the reward they received (i.e., prize draw participation or monetary compensation). Figure 7.16 visualizes the distribution of session durations in seconds (s). The domain-specific box-whisker plots show that participants in the movie experiment, on average, took the longest to complete the study. The mean (M) duration in the movie sessions was $1448s$ (standard deviation [SD] = $1299s$), which corresponds to approximately 32 minutes (min). In other studies, such as in the experiments on book or movie RS, participants needed less time for completion (book RS: $M = 1077s$ [$\sim 18min$], $SD = 604s$; movie RS: $1448s$

[$\sim 24min$], $SD = 1080s$). Although the movie experiment was not comprised of more retrieval tasks, it might have required longer processing times due to the larger dataset. It is also assumed that, since the study on movie RS was the first in the sequence of the multimedia experiments, it may have caused more interest among potential participants, because of the topic of the study. The server-side workload may have been increased, such that response times were slowed down. This effect is also illustrated by the comparably high rate of study withdrawals: only 15.9% of started sessions were finished in the movie experiment. The completion rate in other studies was much higher. For instance, in the travel experiment, 42.2% of users, who accessed the start screen, finished the survey.

Despite these findings, further investigations are required to fully assess the SKOSRec engine's performance in a live setting. Due to financial constraints, the web application ran on a hardware infrastructure, which was not equipped to handle numerous concurrent accesses. The effect of prolonged response times may be mitigated by adding more hardware resources (e.g., more GB of RAM or multi-core processors) to prepare the engine for live deployment.

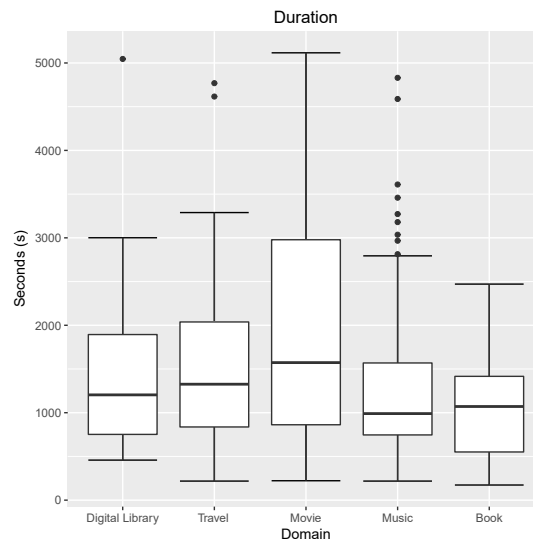


Fig. 7.16: Distribution of session durations in the different domains

Besides server access statistics, it was investigated how participants perceived the general performance of the SKOSRec engine in the execution mode of *on-the-fly retrieval*. The analysis of score values for this retrieval task served two purposes. The first was to ensure that the LOD-enabled recommendation approach produces results that are helpful to consumers. Otherwise, it would be questionable if performance improvements from other retrieval methods (i.e., *constraint-based* or *advanced recommendation requests*) are of actual use to recipients. Slight increases in the quality of a badly performing system may not have a big impact on the advancement of RS research. The second purpose of the first test case (TC1) was to gather baseline performance scores with which other recommendation approaches of the SKOSRec engine could be compared.

Prior to analyzing assessments, it was checked whether the respective Likert items correctly

measured the construct of *perceived usefulness* through reliability testing. For this purpose, Cronbach's α values were determined. As has been mentioned in the previous section, in the studies on DL search and travel RS, the set of Likert items was comprised of two questions, whereas in the subsequent multimedia experiments five questions were utilized to increase α values. Table 7.7 shows that the enhancement improved the reliability of the scale. Since the scores achieved an at least acceptable level of reliability ($\alpha > 0.7$) throughout the experiments, the construct of *perceived usefulness* can be treated as an interval-scaled variable in the entire analysis [238].

Table 7.7: Cronbach's α values for the concept of *perceived usefulness*

Domain	Cronbach α
Digital Library	0.7038
Travel	0.7002
Movie	0.8579
Music	0.8515
Book	0.8949

Besides *usefulness*, user-based *diversity* (*divU*) scores were also measured by multiple Likert items. However, Cronbach's α values for this concept did not reach acceptable levels. In order to avoid that valuable user assessments remained unused, agreements to the statement „The recommendations of the list are diverse “ were taken into account as an ordinal variable.

Once, Cronbach's α values had been detected, subsequent test were carried out. The first analysis concerned the general quality of recommendations resulting from simple *on-the-fly retrieval*. Figure 7.17 depicts the distribution of score values for the concept of *perceived usefulness* throughout the different domains. The diagram shows that participants mostly perceived the results as useful. The vast majority of subjects in the studies on travel and multimedia RS either clicked 5 („strongly agree“), 4 („agree“) or at least 3 („neutral“), when confronted with positive Likert statements on recommendation quality. Thus, it can be concluded that *on-the-fly suggestions* can assist users in searching for interesting items. Despite the positive findings in the DBpedia experiments, the diagram reveals a mediocre performance of the approach in the DL domain. Here, most participants were not quite as convinced of the quality of suggestions. Numerous subjects did not agree to the positive statements on the results. One reason for this outcome might be that EconStor LOD offers too little bibliographic data in order to meet highly specialized information needs. Another reason can be that researchers are proficient in searching through concept descriptors and were therefore already familiar with the papers that were presented to them. This argument is backed up by the results of the demographics section, which indicate that most of the participants in the DL experiment perceive the search for scientific publications as rather easy. Hence, because of the effectiveness of existing tools in this domain or the proficiency of users, a simple LOD-enabled recommendation request might not improve the search experience. However, in terms of quantity, the DL sample represents only a

small proportion of the collected user assessments.

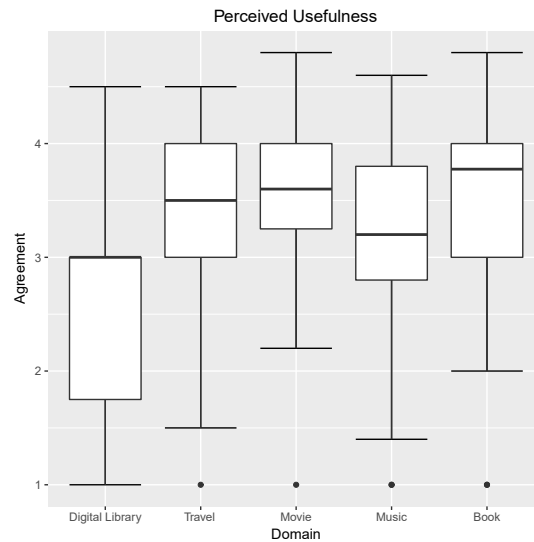


Fig. 7.17: User agreement to positive statements on the *perceived usefulness* of *on-the-fly recommendations*

The overall performance results confirm that the *on-the-fly approach* represents a viable retrieval strategy. Tables 7.8 and 7.9 show the mean², standard deviation and number of participants for each performance dimension and metric. It can be seen that users assessed the engine's performance in almost each quality dimension to be above mediocrity. Despite the positive user evaluations with regard to *perceived usefulness*, related quality aspects, such as *accuracy*, *novelty* and *diversity* received good assessments as well. For instance, in each domain *mean relevance scores (mrs)* were higher than 50 score points. It means that participants were often convinced of the relevance of the results. This outcome is also supported by fairly high *precision* scores. Given these positive user assessments with regard to item relevance, it seems to be justified to consider the mean of result list *sizes* as an indicator of *recall*. The engine was able to generate the required number of recommendations in almost every test case (Tab. 7.8). Aside from high *precision* and *recall* values, the ranking performance of the approach (*ndcg*) also achieved above-average results. However, it must be noted that recommendation lists were not particularly long in the experiments, as result sets contained a maximum number of 10 items in the DBpedia experiments and a maximum number of 6 items in the DL experiments.³ It is assumed that high *ndcg* values are even more useful for larger recommendation lists.

The *on-the-fly retrieval approach* was also tested with regard to the *novelty* dimension. Table 7.9 shows the mean novelty scores (*nv*) for this recommendation approach. The values indicate that the majority of items in a result list were deemed to be novel in the experiments on DL,

²In case of the user-based *diversity score (divU)*, *M* denotes the median, because of the ordinal scale of the variable.

³The lower maximum number of suggestions in the DL experiment was chosen because the author assumed that the assessment of the relevance of scientific publications would take up more time than judging the quality of multimedia or travel destination recommendations.

Table 7.8: Performance results of regular *on-the-fly* retrieval, part I

Dimension (Metric)	Domain	Results		
		M	SD	N
Accuracy (<i>size</i>)	DL	5.74	0.44	39
	Travel	10.00	0.00	103
	Movie	10.00	0.00	50
	Music	10.00	0.00	53
	Book	9.80	1.40	51
Accuracy (<i>mrs</i>)	DL	50.25	23.29	39
	Travel	56.49	18.55	103
	Movie	62.25	19.15	50
	Music	55.70	21.13	53
	Book	57.47	15.55	50
Accuracy (<i>prec</i>)	DL	0.59	0.32	39
	Travel	0.69	0.29	103
	Movie	0.78	0.27	50
	Music	0.67	0.33	53
	Book	0.73	0.24	50
Accuracy (<i>ndcg</i>)	DL	0.89	0.11	39
	Travel	0.86	0.13	103
	Movie	0.90	0.16	50
	Music	0.88	0.16	53
	Book	0.89	0.10	48

travel and book RS, whereas in the experiments on movie and music RS only one third of the recommendation list was unfamiliar to users. In contrast to that, content-based diversity scores (*divC*) were rather high in each domain. Hence, it can be concluded that items in a result set usually have a low rate of matching SKOS annotations. This is an interesting finding, given that recommendations are based on concept descriptors that can be found in the user profile. In addition to content-based scores, user assessments of recommendation list *diversity* (*divU*) were also taken into account. The median level of agreement to the diversity-related Likert item stating that recommendations are diverse was 3 („neutral“) in the multimedia domains and 4 („agree“) in the DL and travel domain.

In summary, the baseline approach achieved fairly good performance results. The findings demonstrate that LOD-enabled recommendations can potentially enhance a user’s search experience. They also allow for subsequent tests of more advanced retrieval techniques, which can be compared to this standard method.

In addition to the evaluation of the engine’s baseline performance, interdependencies between the different performance metrics were measured. The author investigated whether the scores in the quality dimensions of *accuracy*, *novelty* and *diversity* had an impact on the *perceived usefulness* of recommendations, as was claimed by Pu et al. [193]. For this purpose, one-sided Spearman’s correlations were run. Table 7.10 shows the outcome.

Table 7.9: Performance results of regular *on-the-fly* retrieval, part II

Dimension (Metric)	Domain	Results		
		M	SD	N
Novelty (<i>nv</i>)	DL	0.71	0.32	36
	Travel	0.60	0.34	101
	Movie	0.36	0.31	50
	Music	0.36	0.39	52
	Book	0.61	0.29	47
Diversity (<i>divU</i>)	DL	4	-	39
	Travel	4	-	102
	Movie	3	-	50
	Music	3	-	53
	Book	3	-	50
Diversity (<i>divC</i>)	DL	0.64	0.15	39
	Travel	0.80	0.16	103
	Movie	0.87	0.12	50
	Music	0.86	0.10	50
	Book	0.92	0.08	50
Usefulness	DL	2.55	1.00	39
	Travel	3.34	0.85	103
	Movie	3.55	0.73	50
	Music	3.18	0.85	53
	Book	3.43	0.81	50

In a subsequent two-sided correlation analysis, dependencies among presumably related quality metrics in the *accuracy* (*mrs*, *prec*, *ndcg*) and *diversity* (*divC*, *divU*) dimensions were investigated. The results are listed in Table 7.11. This was done to identify variables that can be excluded from subsequent statistical tests because of their high correlation, since a reduction of statistical comparisons decreases the probability of making a type I error (i.e., detecting a significant difference when there is none).

It was also decided to counterbalance effects of multiple testing with an α -level adjustment. Whenever significance tests were conducted for the results of the web experiments, the false discovery rate (FDR) was applied. Thus, the author limited the probability of conducting a type I error, while controlling for type II errors (i.e., overseeing an effect in the data) as well [31].

Table 7.10: Spearman's ρ for evaluation metrics correlation with *perceived usefulness*, TC1

Domain	Accuracy				Novelty <i>nv</i>	Diversity	
	<i>size</i>	<i>mrs</i>	<i>prec</i>	<i>ndcg</i>		<i>divU</i>	<i>divC</i>
DL	-	0.493***	0.355*	-	-	-	-
Travel	NA	0.346***	0.287***	-	-	0.406***	-
Movie	NA		-	-	-	-	-
Music	NA	0.651***	0.442***	-	0.409***	-	-
Book	NA	-	-	-	-	-	-

Table 7.11: Spearman's ρ for dependencies among evaluation metrics, TC1

Domain	Accuracy			Diversity
	$\rho_{mrs,prec}$	$\rho_{mrs,ndcg}$	$\rho_{prec,ndcg}$	$\rho_{divU,divC}$
Digital Library	0.887***	0.515***	0.580***	-
Travel	0.801***	0.504***	0.702***	-
Movie	0.636***	0.135*	0.637***	-
Music	0.559***	-	0.637***	0.376***
Book	0.415**	0.328*	0.741***	-

A-level adjustments were applied for the correlation tests with the FDR rate set to 0.05. Tables 7.10 and 7.11 show correlations with adapted significance levels. It can be seen that, despite the FDR adjustments, some significant dependencies were detected ($*p < 0.05$, $**p < 0.01$, $***p < 0.001$). The results of Table 7.10 confirm the findings from previous research on RS evaluation that *accuracy*, *novelty* and *diversity* have an impact on the *perceived usefulness* of recommendations [194]. As it was claimed by Pu et al., *accurate* suggestions are most likely to be perceived as useful by consumers. The *mrs* metric proved to be the best predictor of *usefulness* among the set of *accuracy* metrics. It is closely followed by *precision* (*prec*). This finding also occurred in subsequent test cases (TC2-TC4) of the study series (Appendix B.3, Tabs. B.1, B.3 and B.5), where scores for *mrs* and *prec* were moderately correlated with the concept of *perceived usefulness*. Additionally, the correlation analysis revealed that increased *ndcg* values only have a weak direct impact on recommendation quality. They are highly correlated with *precision* scores, which are in turn strongly dependent on *mrs* (Tab. 7.11, Appendix B.3, Tabs. B.2, B.4 and B.6). Therefore, it seems reasonable to reduce the set of *accuracy* metrics to *mrs* scores, since the metric has the highest impact on *perceived usefulness*. Additional metrics would not add explanatory power, but only increase the number of tests to be performed. Hence, the *accuracy* metrics *prec* and *ndcg* will be excluded from statistical comparisons for subsequent test cases (TC2-TC4) (Sects. 7.5-7.6). The offline experiments, that were conducted to evaluate the *flexible similarity detection* approach (TFlex) (Sect. 7.4) are an exception to this rule, since no *mrs* scores could be collected in the simulations. For these experiments, the author utilizes *prec* values instead. In terms of *recall* (as indicated by *size*), participants gave higher *usefulness* scores for lists that contained more items (Appendix B.3, Tabs. B.1, B.3 and B.5). Hence, the *size* should be included in subsequent analyses.

Aside from the *accuracy* dimension, the relative importance of *novelty* and *diversity* was evaluated as well. The analysis showed that increased scores in these dimensions can improve recommendation quality. However, in comparison to *accuracy*, the correlation values were lower and only occurred occasionally (Tab. 7.10, Appendix B.3, Tabs. B.1, B.3 and B.5). It also became clear that *diversity* is slightly more relevant for recommendation quality than *novelty*.⁴

⁴Compare results for *nv* and *divU* in TC2, TC3 and TC4 (Appendix B.3, Tabs. B.1, B.3, B.5). In each test case, more one-sided dependencies were identified between *divU* and *perceived usefulness* than between *nv* and *perceived usefulness*

This is in contradiction to the results of Pu et al., who found *novelty* to be more relevant than *diversity* [194]. The difference in the relative ranking of *diversity* and *novelty* may be due to the fact that *novelty* was measured differently in the two empirical studies. Pu et al. applied Likert-type questions to let users assess the entire recommendation list. The present study, on the other hand, surveyed this aspect at the item level, since this method enables a more fine-grained evaluation of *novelty*. It remains open whether one of these methods is superior to the other.

Another interesting finding of the correlation analysis is that user-based *diversity* scores (*divU*) are a more reliable predictor of *usefulness* than content-based *diversity* scores (*divC*). Significant positive correlations occurred in none of the test cases in terms of *divC* (Tab. 7.10, Appendix B.3, Tabs. B.1, B.3, B.5). To the contrary, in TC3, even a negative correlation was found. On the other hand, the two diversity metrics (*divU* and *divC*) are moderately positively correlated in the music domain (see Tab. 7.11). Given these ambiguous findings and the fact that in offline simulations (TFlex) the researcher had to resort to the *divC* metric anyway the variable is kept in the evaluation.

Upon having identified baseline scores and suitable test variables, the SKOSRec engine's novel recommendation approaches will now be comprehensively evaluated.

7.4 Evaluation of Flexible Similarity Detection

7.4.1 Methods for the Evaluation of Flexible Similarity Detection

The *flexible similarity detection* method was tested in both online and offline experiments. The author chose this mixed method approach due to economic restrictions. In the web-based experiments, users had to carry out a variety of user tasks, starting with the assessment of general recommendation quality (TC1) and ending with the evaluation of other types of recommendation queries (TC2-TC4). Although subjects were given an incentive for participation, there were some limitations with regard to the time span users could be requested to devote to the study. Thus, experiments were carried out on historical datasets, if possible. However, it would not have been feasible to evaluate the SKOSRec engine's ability to generate useful suggestions for customized recommendation requests with historical data. The development of a query language brings it about that user opinions on the language's queries do not yet exist. Offline evaluations, on the other hand, are dependent on session logs of RS. Hence, the performance of the system cannot be assessed for the novel query patterns that the SKOSRec engine provides. However, it was possible to evaluate the *flexible similarity detection* approach (TFlex) with historical data. A suitable offline dataset for the evaluation of the TFlex test case should have mappings between the item IDs and the respective IRI in the LOD cloud. In the multimedia domains, there are some historical RS datasets providing these information (Tab. 7.12).

No datasets exist for the usage scenarios of LOD-enabled DL and travel destination search.

Table 7.12: Historical Dataset Features

Dataset	# Users	# Items	LOD Mappings	# Prefs.
Movie: MovieLens1M [101, 170]	6,040	3,952	3,300 [68, 82, 160]	1,000,209
Music: LastFM [108]	1,892	17,632	11,180 [68, 159]	92,834
Book: LibraryThing	7,279	37,232	8,170 [68, 82, 161]	2,056,487

Therefore, a section of the DL experiment was devoted to the evaluation of suggestions resulting from *flexible similarity detection* utilizing the STW SKOS vocabulary. Since the travel experiment focused on the assessment of *advanced query* patterns, there was no extra room for tests of another method, because of the limited attention span of participants. Hence, a section on *flexible similarity detection* was excluded from the travel experiment. This was considered legitimate as the characteristics of the respective SKOS are more decisive for the TFlex test case than the usage scenario. Additionally, the DBpedia SKOS category graph was already covered by the multimedia scenarios.

In case of the multimedia datasets, researchers from the Information Systems Laboratory at the University of Bari have set up links between multimedia items and DBpedia thus facilitating offline evaluations of LOD-enabled recommendation approaches. The collection of mapped datasets comprises the MovieLens1M (movie domain), the LastFM (music domain) and the LibraryThing (book domain) dataset [57]. The collections have been gathered from real-world recommender systems. Table 7.12 lists the number of users and items in each dataset. It also shows, how many LOD mappings could be created.

The author applied the LOD-enabled RS datasets in offline simulation runs to test the *flexible similarity detection* method. First, the datasets were prepared for cross-validation. Preferences from the first 100 users of each collection were divided into 5 equally large subsets on which simulations were performed. In each run, one fifth of the preference information from users was applied as test and four fifths as training data. The simulations were implemented in Java code (see Appendix B.1). The application processed user profiles with the SKOSRec engine and compared the predicted values with the actual feedback data stated in the test set thereby determining the degree of concordance between the engine's predictions and the user's true preferences. By this means, accuracy (*prec*, *rec*) values were calculated. Additionally, the test measured the content-based diversity (*divC*) of recommendation lists and calculated item novelty scores based on the popularity-based approach (*nov_{pop}*).

The different metrics were used to compare the *flexible similarity detection* method with the baseline method of simple *on-the-fly retrieval* (i.e., exact concept matching) and a concept expansion on `skos:broader` links. As outlined before, it was assumed that users can profit from the flexible approach by changing the composition and ranking of the recommendation list to explore knowledge graphs better. The script also measured the overlap between two rec-

ommendation lists (A and B) resulting from different retrieval approaches by the Jaccard index (JI) (Eq. 7.12) to test whether this assumption was correct [237].

$$JI(A, B) = \frac{|A \cap B|}{|A \cup B|} \quad (7.12)$$

It is further hypothesized that *flexible similarity detection* is better suited for *on-the-fly retrieval* than a simple expansion on `skos:broader` links.

The author applied the *conceptSim_{hyp}* metric by Stankovic et al. to calculate concept-to-concept similarity values on the DBpedia SKOS graph. As has been mentioned in Section 6.5 this measure relies on a pondering function that decreases the similarity value the further away the concept is located from the initial concept [233]. Stankovic et al. propose an exponential function for this purpose (see Eq. 6.16 of Sect. 6.5). However, the researchers do not specify the exact value for the parameter γ , which regulates how strongly the similarity decreases with an increasing shortest path distance between two concepts. Since the authors do not provide any empirical evidence regarding the best parameter, preliminary simulations had to be run to identify the optimal configuration. Fig. 7.18 shows the *f1* scores indicating recommendation quality for different γ values in the multimedia domains.

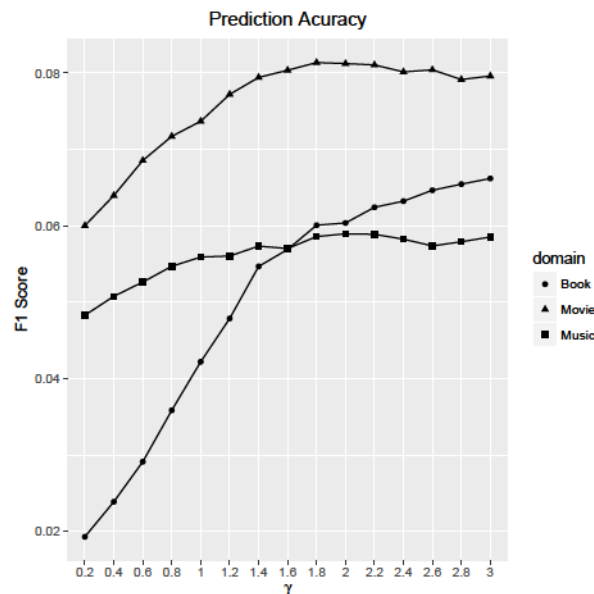


Fig. 7.18: Results of the experiments on parameter configuration (γ parameter)

After determination of the optimal γ parameter, the author tested configurations with varying shortest path distances from the respective seed annotation. Thus, it was investigated whether adjacent concepts ($d(u, v) = 1$, d1 in Fig. 7.19) or the wider neighbourhood of an annotated seed concept ($d(u, v) = 2$, d2 in Fig. 7.19) should be taken into account (see Eq. 6.16 of Sect. 6.5). The simulations did not include longer path distances, since, with each increment of the distance value, the amount of data to be processed rose by a multitude due to the cyclic

dependencies of the SKOS graph. Two-step distances were deemed to be manageable in the context of a real-world ad-hoc recommendation scenario. Figure 7.19 shows the results of preliminary distance-based simulations runs. It can be seen that the incorporation of a wider concept neighborhood ($d(u, v) = 2$) only increased performance results in the music domain.

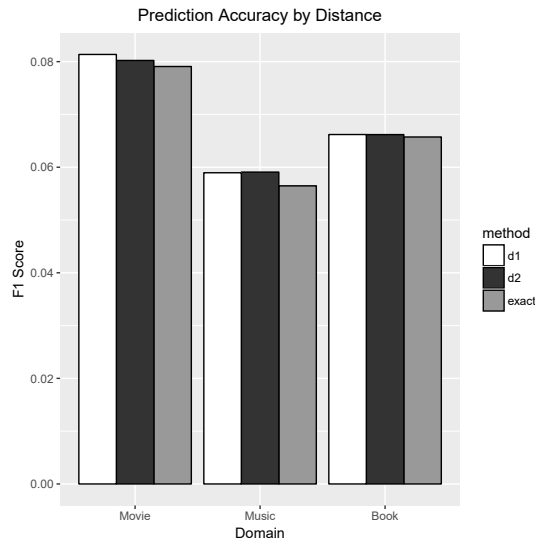


Fig. 7.19: Results of the experiments on parameter configuration (distance parameter)

These pretests identified the best parameters for the *conceptSim_{hyP}* metric to be used to compare the *flexible similarity detection* method with the baseline approach in offline experiments on multimedia RS.

In the DL domain, users evaluated the concept expansion method in the online experiment (i.e., TFlex test case). Because of the restricted time frame in the live study, the author reduced the number of test conditions by omitting the method of `skos:broader` concept expansion. Instead, the experiment focused on the identification of the best concept-to-concept similarity configuration, which was then compared to the approach of simple *on-the-fly retrieval* at a later stage of the analysis (see Subsect. 7.4.2). While this approach led to a loss of information, it was nevertheless necessary due to the limited attention span of participants. Hence, in the DL domain, it was not investigated whether *flexible similarity detection* can outperform `skos:broader` concept expansion, but only examined whether the proposed method is in and of itself a viable alternative retrieval strategy to exact concept matchings. Future experiments will have to clarify if there exist performance differences between the `skos:broader` and the similarity-based concept expansion method in the context of this usage scenario.

While the multimedia simulation runs utilized the DBpedia SKOS graph, the *flexible similarity detection* method in the DL experiment was tested with the STW vocabulary. As has been mentioned before, the STW graph contains transitive relationships and no cyclic dependencies. Thus, it resembles a DAG-like structure that is suited for application of a standard concept-to-concept similarity metric based on the information content of the *MICA* [146]. For the STW, concept specificity was determined according to the number of descendants a concept

subsumes [221] (see Eqs. 6.13 and 6.14 in Sect. 6.5). The metric was chosen because it is well suited to measure corpus-independent concept relatedness in a directed acyclic graph exploiting `skos:broader` and `skos:narrower` links. For each concept pair of the STW thesaurus concept similarities were calculated with the help of the Semantic Measures Library [100]. The values were saved in the default LOD repository to be ready for use during the study. In this way, the web application could quickly present suggestions resulting from *flexible similarity detection* to participants. In the DL experiment, users were asked to assess the quality of three differently ranked lists. As in TC1, users stated a publication that represented their research interests well and the SKOSRec engine processed this information accordingly. The system generated each list with a different inter-concept similarity threshold ($\epsilon \in \{0.5, 0.75, 1.0\}$). Once automated suggestions had been generated, participants were asked to judge the quality of the lists individually. To avoid order effects, the sequence of the recommendation lists resulting from the different methods was chosen randomly in each user session. The user interface for the separate assessments was similar to the evaluation screen used in TC1. Subjects stated their opinions on the relevance of suggestions both on the item level (through the relevance slider and *novelty* statements) and for the recommendation list as a whole. After they had rated each result set, they were asked to rank the lists according to *diversity* and *perceived usefulness* (see Fig. 7.20).

Fig. 7.20: DL - Results, TFlex (page 6)

The similarity configuration with ϵ set to 0.5 achieved, on average, higher ranks regarding usefulness than the setup with $\epsilon = 0.75$.⁵ Hence, the author selected the results of the *flexible similarity detection method* obtained with $\epsilon = 0.5$ to be compared with exact concept matching ($\epsilon = 1.0$). As in the offline simulation runs, concordance between recommendation lists was measured by the Jaccard index in the backend of the application.

The following section will present the results of the statistical analysis comparing simple *on-the-fly retrieval* with the method of *flexible similarity detection* in the different usage scenarios.

⁵In total, the approach with the lower similarity threshold outperformed the other one in 23 out of 39 user sessions

7.4.2 Results of the Evaluation of Flexible Similarity Detection

The main results of the offline simulation runs for the usage scenarios of movie, music and book RS are depicted in Figures 7.21 and 7.22. Comparisons can be made between exact concept matching (*on-the-fly retrieval*), `skos:broader` expansion and the *flexible similarity detection* approach applying the best-performing similarity configuration that was determined by preliminary simulation runs (Subsect. 7.4.1). It can be seen from these results that the *flexible similarity detection* method does, in fact, improve *accuracy* values. This approach almost always achieved the best *precision* (*prec*) and *recall* (*rec*) scores, judging from the diagrams. The sole exception is the *precision* value in the book domain which indicates a slightly weaker performance of the method in comparison to the approach of exact concept matchings. One can further draw the conclusion that a concept expansion on `skos:broader` links is not particularly well suited for *on-the-fly recommendation* scenarios in the multimedia domain since the diagrams show a weak performance of this approach in the *accuracy* dimension. The scores of the `skos:broader` approach are far lower than the scores of the other two approaches. Tables 7.13-7.15 give an overview of the evaluation results from the simulation runs in the multimedia domains. Table 7.13 lists mean values and standard deviations. The author conducted additional Friedmann tests to verify any differences between the methods. Friedmann tests provide lower statistical power than a repeated measures ANOVA but according to Demšar et al. nonparametric tests should be applied on cross-validated data [63]. In case of significant differences, the ranking of the approaches regarding mean scores is marked accordingly (the best performing approach in bold, the second best in underlined and the worst approach in dotted underlined figures).

Similarly to previous statistical tests, α -levels were adjusted with the false discovery rate. In terms of *precision*, Friedman tests confirmed significant differences in the movie domain and the book domain (Tab. 7.14). In the movie domain, additional Wilcoxon pairwise post-hoc tests (FDR-adjusted) verified the superiority of the *flexible similarity detection* method over the other approaches, whereas in the book domain post-hoc tests only confirmed that `skos:broader` concept expansion performs worse than the other retrieval strategies (Tab. 7.15).

For the performance metric of recall (*rec*), Friedman tests confirmed significant differences in each multimedia domain (Tab. 7.14). Subsequent post-hoc analyses revealed a comparably weaker performance of the `skos:broader` concept expansion method in comparison to the other two recommendation approaches. In the domains of movie and book RS, no significant differences could be identified between exact concept matching and *flexible similarity calculation*. In the music domain, however, the *flexible similarity detection* method performed significantly better than the other two approaches in terms of *recall* (Tab. 7.15).

In summary, these findings indicate a poor performance of the `skos:broader` concept expansion method with regard to *accuracy* values, whereas the other two approaches mostly performed equally well. Additionally, in the movie and music domains significance tests even

confirmed the superiority of the *flexible similarity detection* method with regard to either *precision* (*prec*) or *recall* (*rec*).

In the *novelty* dimension almost no meaningful differences were identified between the three retrieval approaches. Despite the small variations in *novelty* scores that are depicted in Figure 7.22, Friedman tests confirmed their significance only for the movie domain dataset (Tab. 7.14), where Wilcoxon post-hoc analyses revealed that `skos:broader` concept expansion, on average, produced the most novel recommendations, followed by the approach of exact concept matching (Tab. 7.15). The *flexible similarity detection* method, while generating highly precise results, often recommended rather popular items.

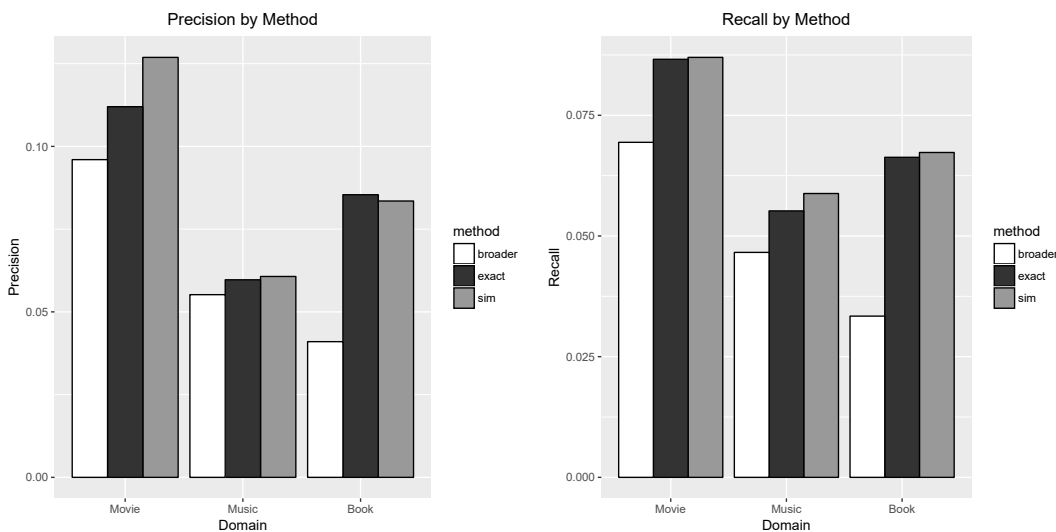


Fig. 7.21: Results of offline simulation runs regarding *accuracy*

For *diversity* scores, Friedman tests confirmed significant differences in each multimedia domain (Tab. 7.14). The ranking of the methods in terms of *divC* scores, as is shown in Figure 7.22, was confirmed by statistical tests in the movie and in the book domain. The increased *divC* scores of the concept expansion methods are not surprising, given the fact that they incorporate additional item features into the similarity calculation process, thereby diversifying recommendation lists.

The results of the simulation runs show that the *flexible similarity detection* method can have a positive effect on recommendation quality. In some domains, the method proved to be significantly better regarding *accuracy* scores. In other cases, *flexible similarity detection* did not lead to any significant improvements of *precision* or *recall* in direct comparison to exact concept matching, but still had a positive impact on *diversity* values. It needs to be kept in mind, however, that the diversification of result lists according to *divC* scores does not necessarily lead to improvements of *perceived usefulness*. Concept expansion on `skos:broader` links seemed to have had the same positive impact on *diversity*, but at the same time led to significantly worse *accuracy* values. Hence, improvements in the *diversity* dimension came at the cost of *precision* and *recall*.

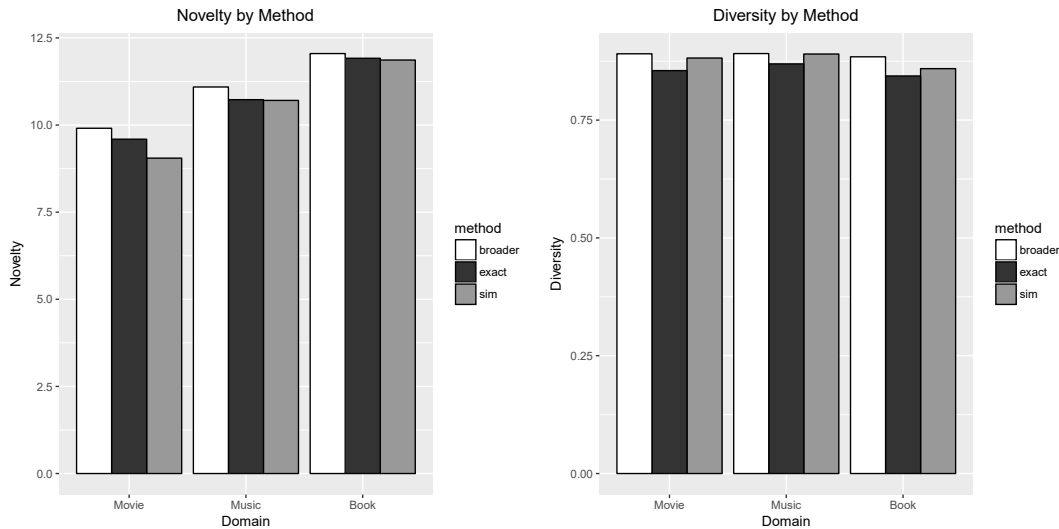


Fig. 7.22: Results of offline simulation runs regarding *novelty* and *diversity*

In summary, it can be stated that the *flexible similarity detection* approach is a viable alternative strategy for LOD-enabled retrieval tasks in multimedia usage scenarios. It diversifies recommendation lists while providing at least the same level of *accuracy* (RQ11). A concept expansion on `skos:broader` links, however, proved to be not as valuable.

Table 7.13: Performance results of the simulation runs (multimedia domain)

Dim. (Metric)	Domain	broader		exact		sim		Sig.
		M	SD	M	SD	M	SD	
Accuracy (<i>rec</i>)	Movie	<u>0.0694</u>	0.0581	<u>0.0866</u>	0.0576	0.0870	0.0638	✓
	Music	<u>0.0466</u>	0.0445	<u>0.0552</u>	0.0529	0.0588	0.0445	✓
	Book	<u>0.0334</u>	0.0458	<u>0.0663</u>	0.0750	0.0673	0.0765	✓
Accuracy (<i>prec</i>)	Movie	<u>0.0960</u>	0.1084	<u>0.1120</u>	0.1046	0.1269	0.1267	✓
	Music	<u>0.0552</u>	0.0566	0.0597	0.0587	0.0607	0.0581	✗
	Book	<u>0.0410</u>	0.0456	<u>0.0854</u>	0.1060	0.0835	0.1021	✓
Novelty(<i>nov</i>)	Movie	9.9085	1.1475	<u>9.5948</u>	0.8020	9.0511	2.2065	✓
	Music	11.0932	3.0087	10.7295	3.7810	<u>10.7076</u>	3.7469	✗
	Book	12.0522	2.1212	11.9175	2.5186	11.8663	2.4642	✗
Diversity (<i>divC</i>)	Movie	0.8907	0.0533	<u>0.8550</u>	0.0533	<u>0.8817</u>	0.0526	✓
	Music	0.8912	0.0450	<u>0.8693</u>	0.1036	<u>0.8902</u>	0.0371	✓
	Book	0.8843	0.1013	<u>0.8437</u>	0.1329	<u>0.8592</u>	0.1291	✓

The claim of the *flexible similarity detection* method as a helpful alternative retrieval strategy is further backed up by the fact that the composition of recommendation lists was different between the approaches throughout the domains. Jaccard indices (JI) were determined for each method pair. Table 7.16 lists the results of these calculations. The mostly low JI values indicate that concept expansion methods change the composition of recommendation lists thereby enabling a more comprehensive exploration of the item space through browsing.

Table 7.16 also depicts the mean Jaccard value for the result sets obtained from *flexible sim-*

Table 7.14: Significant differences according to Friedman tests (multimedia domain)

Dim. (Metric)	Domain	Friedman test
Accuracy (<i>rec</i>)	Movie	$\chi(2) = 53.18, p < 0.001, N = 100$
	Music	$\chi(2) = 25.12, p < 0.01, N = 100$
	Book	$\chi(2) = 80.14, p < 0.001, N = 100$
Accuracy (<i>prec</i>)	Movie	$\chi(2) = 22.59, p < 0.001, N = 100$
	Book	$\chi(2) = 36.97, p < 0.001, N = 100$
Novelty (<i>nov_{pop}</i>)	Movie	$\chi(2) = 148.34, p < 0.001, N = 96$
Diversity (<i>divC</i>)	Movie	$\chi(2) = 115.74, p < 0.001, N = 100$
	Music	$\chi(2) = 23.38, p < 0.001, N = 94$
	Book	$\chi(2) = 38.21, p < 0.001, N = 100$

Table 7.15: Significant differences between the retrieval strategies as verified by Wilcoxon post-hoc tests (multimedia domain)

Dimension (Metric)	Domain	exact vs. broader	sim vs. exact	sim vs. broader
Accuracy (<i>rec</i>)	Movie	$Z = -5.55, p < 0.001$	-	$Z = -9.47, p < 0.001$
	Music	$Z = -3.19, p < 0.01$	$Z = -2.63, p < 0.05$	$Z = -4.46, p < 0.001$
	Book	$Z = -6.54, p < 0.001$	-	$Z = -6.72, p < 0.001$
Accuracy (<i>prec</i>)	Movie	$Z = -3.86, p < 0.001$	$Z = -3.40, p < 0.01$	$Z = -4.68, p < 0.001$
	Book	$Z = -5.76, p < 0.001$	-	$Z = -5.98, p < 0.001$
Novelty (<i>nov_{pop}</i>)	Movie	$Z = -3.54, p < 0.001$	$Z = -2.29, p < 0.05$	$Z = -4.20, p < 0.001$
Diversity (<i>divC</i>)	Movie	$Z = -7.84, p < 0.001$	$Z = -8.31, p < 0.001$	$Z = -3.57, p < 0.001$
	Music	$Z = -4.08, p < 0.001$	$Z = -4.68, p < 0.001$	-
	Book	$Z = -4.63, p < 0.001$	$Z = -5.97, p < 0.001$	$Z = -3.58, p < 0.001$

Table 7.16: JI of recommendation lists for Tflex

Domain	exact vs. broader	sim vs. exact	sim vs. broader
Movie	0.39	0.56	0.39
Music	0.00	0.00	0.33
Book	0.26	0.78	0.26
Digital Library	-	0.26	-

ilarity detection and on-the-fly retrieval (i.e., exact concept matching) in the DL experiment. As it was the case for offline simulation runs, the JI turned out to be fairly low in this study as well. Additionally, Table 7.17 provides an overview of the main findings from the statistical comparisons between performance scores of exact concept matching and flexible similarity detection from the DL experiment. Pairwise t-tests were conducted for interval-scaled variables (*size*, *mrs*, *nv*, *divC*), among whom only content-based diversity scores (*divC*) were found to be significantly different ($t(38) = -5.43, p < 0.001$), with the flexible similarity method achieving higher diversity values, than exact concept matching. Again, this is not surprising given the altered retrieval strategy of concept expansion. Wilcoxon pairwise comparisons were applied to investigate whether one of the approaches was better ranked than the other one in terms of

diversity (*divU - rank-based*) and general usefulness (*rank-based*).⁶ For each metric, the mean scores of the better performing approach are either underlined (for non-significant differences) or shown in bold figures (for significant differences). In case of the rank-based measures, the median of both performance metrics was lower for exact concept matching, whereas the non-parametric test did not verify any systematic differences.

Given these findings in the DL experiment, it is concluded that the *flexible similarity detection* approach can be an alternative access strategy in this domain when regular results do not offer any interesting suggestions. Performance scores indicate that users might be able to find additional recommendations of comparable quality from a topically more diverse result list.

Besides testing the SKOSRec system in *flexible similarity detection* mode, the author evaluated the engine concerning its ability to provide useful results, when users formulate specific constraints (TC2). The methods and results for this test case will be presented in the following section.

Table 7.17: Evaluation scores for *on-the-fly* (exact) vs. *flexible similarity detection* (DL domain)

Dimension (Metric)	exact		sim		N
	M	SD	M	SD	
Accuracy (<i>size</i>)	6	-	6	-	39
Accuracy (<i>mrs</i>)	<u>48.99</u>	21.75	47.60	22.68	39
Accuracy (<i>prec</i>)	<u>0.57</u>	0.48	0.51	0.32	39
Novelty (<i>nv</i>)	0.68	0.35	<u>0.71</u>	0.34	32
Diversity (<i>divU - rank-based</i>)	<u>1</u>	-	2	-	39
Diversity (<i>divC</i>)	0.62	0.11	0.76	0.16	39
Usefulness (<i>rank-based</i>)	<u>1</u>	-	2	-	39

7.5 Evaluation of Constraint-based Recommendation Retrieval

7.5.1 Methods for the Evaluation of Constraint-based Recommendation Retrieval

Constraints serve as filters on the set of relevant resources before the process of similarity calculation is started. In this context, TC2 addressed two crucial aspects. The first aspect concerned the engine's ability to improve *usefulness* through filter conditions (RQ12). For these comparisons, an appropriate experimental setting had to be applied. The author utilized a within-subjects design, since a large sample that would have been required for a between-subjects experiment could not be generated within the study series of the thesis. Additionally, customized

⁶Note that in the case of rank-based measures, lower figures indicate a better performance

recommendation requests are highly individual. Even small variations in the experimental setting could cause substantial differences in performance evaluations [29]. Hence, to keep the extent of variations even lower, comparisons between simple *on-the-fly* and *constraint-based recommendations* were based on the same profile for each user. Section 7.3 outlined how users generated a personalized profile through an AJAX-based autocomplete form in TC1. This profile served as a starting point for filtered retrieval. After participants had completed the first section of the web experiment (user profile generation, evaluation of results originating from TC1), the second section started (TC2), in which the web application showed the recently generated profiles to users and asked them to provide an additional filter condition. Figure 7.23 depicts an example web form for a *constraint-based recommendation* request in the music domain. It comprises the user profile and an additional text field, where users stated their filter condition. Afterward, the engine applied this filter condition on the set of potential recommendation results before similarity calculation was started. For each usage scenario, different types of constraints were available. Regarding suitable filter options, the author sought to achieve a balance between assisting users in finding filter conditions that would adequately represent their information needs, while keeping the variability among domains as low as possible.

Q8. Test Case II

In this test case you can filter recommendations according to your interests. You see the music acts you have stated as your preferences in the last section (Q7). Now, you are able to refine these preferences with a condition. Please choose either a subject (e.g. Folk singers) or a genre (e.g. Soul music) that fits your interests. For the recommendation algorithms to work, we kindly ask you to only select conditions that appear in the auto-complete list of the "Filter" field.

Favorite music acts

- The Police
- Sting (musician)
- Bono

Filter

Subject

- British songwriters
- British singer-songwriters
- Songwriters
- English songwriters
- Argentine songwriters
- Jazz songwriters
- Japanese songwriters
- French songwriters

Fig. 7.23: Music - User profile generation, TC2 (page 5)

Therefore, the specificities of each scenario and the availability of the respective data sources were determined. For instance, for the multimedia domains (movie, music, and books) it was assumed that users would like to filter recommendations according to the specific genre of the item. In the music domain, the DBpedia dataset contains this kind of information. It can be re-

trieved through the property `dbo:genre`. However, in the movie and book domains, DBpedia does not provide this data. Many LOD resources which represent books were not assigned a genre. In the movie domain, a genre property is missing entirely and genre-specific information can often be only found in the subject categories describing the movie. Hence, in contrast to the experiment on music RS, participants of the book and movie RS experiments could not filter their recommendation lists by genre. On the other hand, the property `dc:subject` could be applied as a filter in each multimedia domain. Since the subject property often links to many informative features of multimedia items (e.g., release period, geographic or content information), the author decided that this feature type represented a powerful filter dimension. It would have been possible to enable users to state a specific person (e.g., an actor, director or an author) as an additional refinement option as well. However, this kind of information is not available for every multimedia item in the DBpedia dataset [59]. Hence, such a filter option would have unnecessarily excluded items, due to missing information, which in turn could have led to biased evaluation results.

The EconStor dataset provides author information for almost all its publications [79]. This fact as well as the common practice of scientists to (co-)author up to hundreds of papers, led to the assumption that the author filter was a helpful option besides the subject filter in the DL experiment. For the domain of scientific publication retrieval, it was also hypothesized that an option to filter recommendation results according to the series of a paper (e.g., a journal or a working paper series), would be useful as well. Since series often release papers in a certain subdomain of a scientific field, they might help to narrow down the search. For this reason, users could filter recommendations by specifying a series title. In the experiment on travel destination search, participants had fewer options. The web interface had two filters: a subject and a location-specific filter. The subject filter enabled users to state their preferences about the characteristics of a location (e.g., being a nature reserve). The location-specific filter, on the other hand, gave participants the option to specify, where the desired destination should be located. However, DBpedia sometimes lists location-specific information in the subject category as well. Therefore, in the backend of the application, user filters were rewritten into a pattern, which comprised the union of the two graph patterns representing the filter options. Each sub-pattern contained the constraint specified by the user. By this means, the query engine could retrieve all relevant resources from the repository. However, the author decided against joining the two filter categories in the frontend to avoid any confusion [59]. Table 7.18 lists the filter options and the SPARQL graph patterns representing them for each web experiment. In addition to the constraints, the graph patterns also specified the corresponding item types (`?item rdf:type <ITEM_TYPE>`) to ensure that only items from the particular usage scenario were presented. The author omitted these triple patterns in Table 7.18 for brevity reasons.⁷

⁷ PREFIX econAuth: <<http://linkeddata.econstor.eu/beta/resource/authors>>

Table 7.18: SKOSRec filter options for constraint-based recommendation requests

Experiment	Filter	Simple Graph Pattern	Expressive Graph Pattern
Dig. Library	Author	<code>?item dc:creator ?p . FILTER(STRSTARTS(STR(?p), "econAuth:<CONSTRAINT>"))</code>	N.A.
	Subject	<code>?item dc:subject stw:<CONSTRAINT> .</code>	<code>{?item dc:subject ?subConcept . ?subConcept skos:broader* stw:<CONSTRAINT> .} UNION {?item dc:subject ?subConcept . ?subConcept skos:related stw:<CONSTRAINT> .}</code>
	Series	<code>?item dct:isPartOf econColl:<CONSTRAINT> .</code>	N.A.
Travel	Subject/ Location	<code>{?item dbo:isPartOf dbr:<CONSTRAINT> . } UNION {?item dc:subject dbc:<CONSTRAINT> .</code>	<code>{?item dbo:isPartOf dbr:<CONSTRAINT> . } UNION {?location rdfs:subPropertyOf dul:hasLocation . ?item ?location dbr:<CONSTRAINT> . } UNION {?item dc:subject ?subject subject skos:broader{,2} dbc:<CONSTRAINT> . }</code>
	Subject	<code>?item dc:subject dbc:<CONSTRAINT> .</code>	<code>?item dc:subject ?subject subject skos:broader{,1} dbc:<CONSTRAINT> .</code>
Movie	Subject	<code>?item dc:subject dbc:<CONSTRAINT> .</code>	<code>?item dc:subject ?subject subject skos:broader{,1} dbc:<CONSTRAINT> .</code>
	Subject	<code>?item dc:subject dbc:<CONSTRAINT> .</code>	<code>?item dc:subject ?subject . subject skos:broader{,1} dbc:<CONSTRAINT> .</code>
Music	Genre	<code>?item dbo:genre dbr:<CONSTRAINT> .</code>	<code>{?item dbo:genre dbr:<CONSTRAINT> . } UNION {?item dbo:genre ?subGenre . dbr:<CONSTRAINT> dbo:musicSubgenre subGenre . }</code>
	Subject	<code>?item dc:subject dbc:<CONSTRAINT> .</code>	<code>?item dc:subject ?subject subject skos:broader{,1} dbc:<CONSTRAINT> .</code>

Besides simple graph patterns, the table also depicts the *expressive* versions of the filter constraints. As has been previously pointed out, *expressive filter* requests can be applied to overcome data quality issues and can reveal hidden information in the data that might be helpful for the user. Consequently, aside from the general effectiveness of *constraint-based retrieval*, another fundamental research question of TC2 concerned the question whether *expressive constraints* can boost recommendation quality even further (RQ13). Hence, apart from executing a *constraint-based* workflow with a *simple filter* upon receiving a participant's request, the SKOSRec engine generated suggestions with the help of the *expressive filter* as well. This second procedure, while still applying the same user constraint as the simple execution mode (i.e., <CONSTRAINT>) utilized an expressive graph pattern to include additional LOD resources in the result set. In cases, where Table 7.18 does not show an advanced graph filter the author was not able to identify appropriate patterns in the corresponding datasets. Whenever there was a suitable graph pattern available, participants received two separate recommendation lists. One list contained results based on *simple filtering*, and the other showed suggestions for the *expressive constraint*.

However, in the DL experiment, only recommendations resulting from *simple filtering* were shown to participants. This approach was chosen to keep the necessary effort in this study at a reasonable level. Aside from assessing the SKOSRec engine's performance regarding *on-the-fly* (TC1) and *constraint-based retrieval* (TC2), subjects in the DL experiment had to work on an additional test case (TFlex). Since participants had carried out time-consuming evaluations by the time they reached the web form of TC2, the required effort in this section was limited to a minimum to prevent study withdrawals. Hence, in the *constraint-based* section, DL users solely assessed a single recommendation list (i.e., the one resulting from *simple filtering*). Therefore, the DL experiment can only partly answer the research question as to whether the application of constraints can improve recommendation quality. Although this approach imposed some minor limitations on the generalizability of evaluation results, it was deemed necessary due to time and economic restrictions. Additionally, the experiments on travel, movie, music and book RS fully investigated the impact of *expressive graph patterns*. Thus, domain-specific variations were still captured in the remaining experiments. Differences between the DL and the other experiments were also partly leveled out by deferred evaluations on user logs after the DL experiment had already been carried out. In these tests, the author reissued the *constraint-based recommendation* requests of DL participants that contained keyword filters ($N = 15$) with an expanded graph-based filter against the SKOSRec engine. The *expressive* version comprised all ancestors (`skos:broader*`) as well as related concepts (`skos:related`) of the originally specified subject descriptor.⁸ By these means, it was possible to draw at least some conclusions about response rates (see Subsect. 7.5.2). However, the author could only carry out these tests whenever participants had chosen to use the subject filter since EconStor did not contain *ex-*

⁸For a graphic representation of this filter example, please have a look at the respective journal publication related to this research [251].

pressive patterns for the other filter types.

The author utilized the approach of expanded subject filters in the DBpedia experiments as well but adapted it slightly to meet the specificities of the dataset. Since the DBpedia category graph is not a transitive KOS and contains cycles [155], the principles of hierarchical closure could not be applied in the same manner. Hence, expanding user filters on paths of arbitrary length (i.e., on a property path with an asterisk [*]) was not an option, as it would have corrupted the retrieval process. Therefore, property paths were utilized in a predefined range. For instance, in the travel experiment, subjects were restricted to the second sublevel `skos:broader{,2}`. Preliminary tests had revealed that a farther-reaching concept expansion disproportionately slowed down response times, while only considering the immediate children often did not identify all the relevant geographic entities for a particular region. In the multimedia domains, however, initial tests had shown that concept expansion on the first sublevel (`skos:broader{,1}`) sufficed to retrieve most of the relevant LOD resources.

Aside from SKOS-based filter expansion, the author applied further expressive patterns in the experiments on music and travel RS. In the music domain, the genre filter required the exploration of another concept scheme, which also has a hierarchical structure. In DBpedia, music-related subgenres are expressed with the property (`dbo:musicSubgenre`). Nevertheless, as the category graph, the music genre hierarchy contains cycles as well [245]. Hence, the *pressive* genre filter was only expanded to immediate children nodes.

In the travel domain, the extra filter retrieved place-related information. In reference to Example 2 given in Subsection 3.2.4, the author applied a query rewriting rule to obtain the RDFS closure of location-specific sub-property relations that are not materialized in DBpedia. The property `dul:hasLocation` links to many properties, such as `dbo:country`, `dbo:city` or `dbo:state`. Therefore, the *pressive graph pattern* (shown in the fifth row and second column of Tab. 7.18) broadened the scope of the result set, while still capturing the notion of a location-based filter.

In the DBpedia experiments, upon execution of *constraint-based recommendation* requests, users received two recommendation lists (resulting from *simple* and *pressive filtering* accordingly) in a separate evaluation screen (see Appendix B.2, Figs. B.32-B.34). Participants evaluated result sets in the same manner as in TC1. Additionally, for each result set, the screen contained a Likert item asking subjects to give their agreement to the statement: „The filter has improved the recommendation results of list...“ As in TFlex, the sequence of suggestions resulting from the different methods was randomly assigned in each user session to avoid order effects.

7.5.2 Results of the Evaluation of Constraint-based Recommendation Retrieval

Constraint-based recommendations were assessed with regard to *accuracy*, *novelty*, *diversity* and *perceived usefulness*. Additional attention was paid to participants' agreements with the statement: „The filter has improved the recommendation results of the list“. Figure 7.24 shows the distribution of domain-wise agreement ratios to this statement. On average, they were fairly high. Throughout the domains, the vast majority of participants either selected 5 („strongly agree“), 4 („agree“) or 3 („neutral“). Overall, the results indicate that filtered requests are a viable recommendation strategy (RQ12). Hence, the filter often had a positive impact on results.

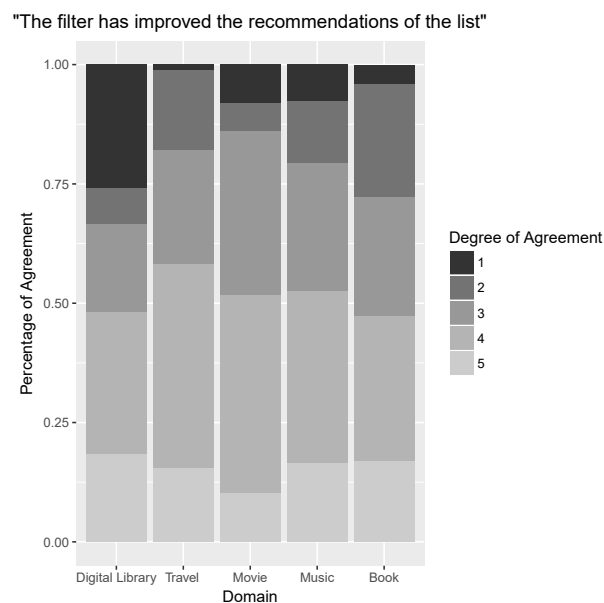


Fig. 7.24: Domain-wise agreement ratios with the statement that the filter has improved the result list

Additionally, the diagram shows that satisfaction was particularly high in the travel domain and quite low in the DL experiment. The low scores in the DL experiment may have occurred because overall *usefulness* scores were lower in this study as well. Thus, the application of a filter on a result list of mediocre quality did not have a big impact. The author gathered responses to the question on improvement from both the *simple* and the *expressive filtering* approaches. Hence, they give clues about the general performance of *constraint-based retrieval*. Since this section of the study revealed the experimental condition to participants (i.e., application of a filter), answers to the Likert statement have to be interpreted cautiously. Results may be slightly biased as participants were able to guess the underlying agenda.

However, these limitations do not exist for comparisons of *usefulness* assessments for *simple* and *expressive filters*. Participants did not know which approach they were evaluating, since the web application randomly assigned the list order.

Figure 7.25 shows the corresponding score values of the baseline approach (*on-the-fly retrieval*) in comparison to the *constraint-based retrieval* methods, i.e. *simple constraint-based* and *expressive constraint-based* (expansion) filtering.

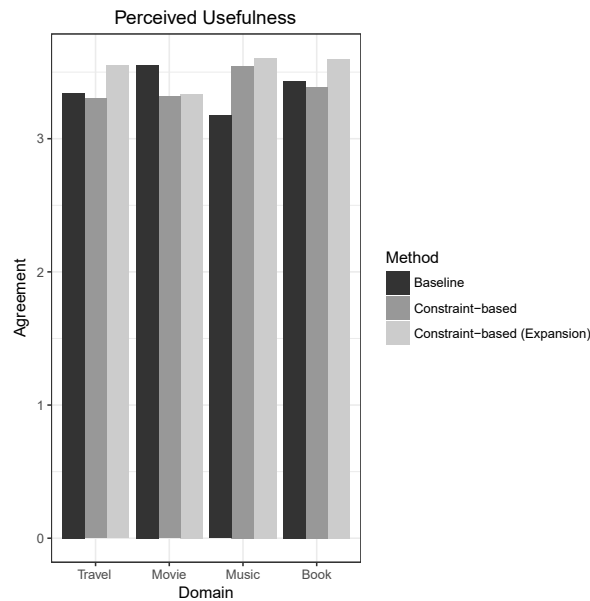


Fig. 7.25: Participants' overall satisfaction with *constraint-based recommendation* retrieval

In almost every domain - except for the movie domain - *expressive filtering* improved user satisfaction in comparison to a *simple filter* (RQ13). The diagram does not depict the scores from the DL experiment, as the study only compared the baseline method with *simple constraints*.

Table 7.19 lists response rates of the two methods. The rate measures the ratio of non-zero result sets among all result sets that were generated during user sessions in this section. Throughout the domains, recommendation lists resulting from *expressive filtering* achieved higher response rates.

Table 7.19: Response rates for TC2

Domain	Constr.-based (Simple)	Constr.-based (Expressive)	N
DL	89%	96%	27
Travel	23%	57%	103
Movie	86%	88%	50
Music	72%	75%	53
Book	73%	76%	51

Additionally, offline simulations determined the response rate in the DL experiment. The author evaluated session logs from TC2 after the DL experiment had already taken place. The post-experimental examination revealed that with the expanded keyword filter, only 1 participant received an empty result set as opposed to 3 participants that did not receive any results during the actual experiment. However, given the small sample, the results from the DL experiment are not as reliable as the findings from the other domains. Besides, nothing can be said about

the relevance of the additionally identified recommendations, since they were not available to participants during the DL experiment. Nevertheless, it is interesting that the findings for this performance indicator are the same in all usage scenarios. In addition to response rates, user assessments in the quality dimensions were compared for the different methods.

Table 7.20 depicts the outcome of this analysis. Significantly higher scores are marked in bold figures. In case statistical tests identified no meaningful variations, the results of the better performing approach are underlined. Again, α -levels were adjusted with the false discovery rate (FDR).

Table 7.20: Performance results of *constraint-based recommendation retrieval*

Dimension (Metric)	Domain	Constr.-based (Simple)		Constr.-based (Expressive)		N
		M	SD	M	SD	
Accuracy (<i>size</i>)	Travel	1.17	2.70	4.52	4.62	103
	Movie	6.72	4.08	<u>7.20</u>	3.92	50
	Music	5.96	4.64	<u>6.36</u>	4.51	53
	Book	4.59	4.50	<u>5.00</u>	4.51	51
Accuracy (<i>mrs</i>)	Travel	<u>66.73</u>	20.51	65.34	22.36	23
	Movie	<u>60.95</u>	22.98	60.29	22.79	43
	Music	<u>63.77</u>	16.05	62.96	16.44	34
	Book	56.17	18.18	<u>56.81</u>	16.81	37
Novelty (<i>nv</i>)	Travel	0.48	0.41	<u>0.61</u>	0.49	21
	Movie	0.44	0.36	<u>0.55</u>	0.80	42
	Music	<u>0.55</u>	0.39	0.52	0.37	38
	Book	<u>0.77</u>	0.34	0.76	0.33	33
Diversity (<i>divU</i>)	Travel	3	-	<u>3.5</u>	-	24
	Movie	4	-	4	-	43
	Music	4	-	4	-	38
	Book	3	-	3	-	37
Diversity (<i>divC</i>)	Travel	0.68	0.18	<u>0.72</u>	0.21	19
	Movie	0.70	0.24	<u>0.73</u>	0.23	41
	Music	0.86	0.13	<u>0.87</u>	0.12	34
	Book	0.84	0.24	<u>0.95</u>	0.63	29
Usefulness	Travel	3.33	0.93	<u>3.50</u>	0.77	24
	Movie	<u>3.32</u>	0.85	3.31	0.86	43
	Music	3.55	0.81	<u>3.64</u>	0.78	38
	Book	3.39	0.83	<u>3.62</u>	1.21	37

According to mean scores, *expressive requests* generated more comprehensive recommendation lists (*size*). This finding is in line with the increased response rates shown in Table 7.19. However, subsequently conducted t-tests confirmed significant differences only for the travel experiment ($t(102) = -7.89, p < 0.001$). In contrast, *precision* scores (*mrs*) were higher in most of the domains (i.e., travel, movie, music) for *simple constraint-based queries*. However, statistical tests did not confirm any systematic differences. The author applied Wilcoxon tests

Table 7.21: JI of recommendation lists for TC2

Domain	Mean Jaccard Index (JI)	SD	N
Travel	0.57	0.47	25
Movie	0.71	0.38	43
Music	0.82	0.35	37
Book	0.92	0.26	35

for *prec* scores and the remaining performance metrics because of the small sample sizes resulting from the low response rates of *simple filtering*.

Regarding *novelty*, none of the approaches was superior. In the *diversity* dimension, *expressive constraints* generated suggestions that were at least as topically diversified as the results from *simple filtering*. Even though no significant differences were identified between the two approaches, increased mean scores (*divU* in the travel domain and *divC* in each domain) indicate a slight superiority of expanded filter requests in this quality dimension. The same applies to *usefulness* scores (mean values were higher in 3 out of 4 domains, when *expressive filtering* was applied). However, these statements are not proven because statistical tests were not significant. In summary, it can be stated that *expressive constraints* improve response rates and *recall*, potentially leading to a slight loss in *precision* scores. It may also be the case that expanded user constraints diversify as well as increase the *usefulness* of recommendation lists. However, these claims are unverified. In contrast, it is safe to say that *expressive filtering* generates results of at least the same quality as *simple filtering* while increasing the number of relevant suggestions. The few differences may be explained by the high JI scores of recommendation lists (see Tab. 7.21). Throughout the domains, result sets were much alike. Given these similarities and the increases in list *sizes* for *expressive filters*, it is supposed that such a constraint produces an enhanced version of the recommendation list resulting from *simple filtering*. Therefore, *expressive filters* should be the default setting in a *constraint-based retrieval* context.

7.6 Evaluation of Advanced Query Patterns

7.6.1 Methods for the Evaluation of Advanced Query Patterns

Aside from processing user constraints as a *prefilter* condition on the set of potentially relevant items, the engine can also apply filters at other stages of the retrieval process. Section 6.6 has introduced the approaches of *preference querying* and *postfiltering* and outlined how different filter options can be combined to formulate *advanced recommendation requests*. The experiments evaluated some of these query patterns. The author formulated example requests for each usage scenario. Thereby, it was attempted to fit the query pattern to common information needs of users in the particular domain. Unfortunately, for the domain of DL search, it was not possible to identify any suitable graph patterns since EconStor is still in beta status [79].

Additionally, EconStor primarily serves as a bibliographic rather than a general reference tool. Hence, the dataset does not contain comprehensive information on entities other than publications. Therefore, it was not feasible to evaluate *advanced recommendation requests* in the context of scientific publication retrieval.

DBpedia, on the other hand, provides comprehensive information on numerous entity types. For each DBpedia experiment, the author devoted at least a single survey section to *advanced queries*. For TC3 in the travel experiment, participants could choose between three *rollup query* patterns. The respective templates were formulated in reference to Example 8 (Sect. 6.6). Users could obtain recommendations for travel destinations based on POIs, they had visited during the stay in another destination. Participants were able to choose between the entity types city, region, and country. Upon entity type selection, the SKOSRec engine generated appropriate suggestions. The three query options were provided to avoid that participants terminated the study early, in case they were not able to state a preferred location for a certain entity type. Figure 7.26 shows the web form from the travel experiment that facilitated the formulation of *advanced rollup requests*.

Home Contact

Q10. Test Case III

In this section you will receive recommendations based on your preferences for certain places. Choose a city (e.g. Berlin), a region (e.g. Tuscany) or a country (e.g. Greece) and provide your favorite places there (e.g. Berlin Wall, Unter den Linden and Museum Island for Berlin). We kindly ask you to only select places that appear in the auto-complete lists as soon as you type.

Please note: When searching for words, mutated vowels should be replaced by interrogation marks.

Travel Destination

City: London ID: London IMG:

Favorite places there

Destination 1 Oxford Street ID: Oxford

Destination 2 Notting Hill ID: Notting_

Destination 3 South Bank

Please note: **South Bank ferry wharf (Brisbane)**
South Bank 1 & 2 is a ferry wharf in the suburb of South Brisbane used by the CityCat on the Brisbane River.

South Bank
The South Bank is an area of Central London, England located immediately adjacent to the south bank of the River Thames. It forms a long and narrow section of riverside development within the London Borough of Lambeth and the London Borough of Southwark. It developed much more slowly than the north bank of the river due to adverse conditions, and throughout its history has twice functioned as an entertainment district, separated by a hundred years of use as a location for industry.

South Bank Parklands
The South Bank Parklands are located at South Bank in Brisbane, Queensland, Australia. The parkland, on the transformed site of Brisbane's World Expo 88, was officially opened to

Fig. 7.26: Travel - User profile generation, TC3 (page 7)

When users had selected an entity type, stated their favorite travel destination (e.g., `dbr:<PREF_CITY>`) and three POIs (e.g., `dbr:<POI_1>`), the application generated a SKOS-Rec query from these parameters. Table 7.22 lists the patterns utilized in the travel experiment. For each entity type option, the table also depicts a corresponding *simple on-the-fly request*. This query was sent to the engine to retrieve baseline recommendations, with which the suggestions resulting from *advanced retrieval* were compared at a later stage of the experiment. The simple query only contained the user's favorite travel destination and the selected entity type option thus omitting information on the POIs. During the travel domain experiment, the engine processed both the simple and the *advanced query* and produced two recommendation lists. As in the previous parts of the experiment, participants assessed the quality of the result sets, which

appeared in random order on the screen.

The multimedia experiments contained a section on *rollup retrieval* (TC3) as well. As the *advanced queries* in the travel experiment, the multimedia requests aggregated similarity scores of sublevel entities through summation, which the engine joined with a *postfilter*. In addition to these features, the pattern contained a *preference query* part. This section identified a set of preferred items based upon a single user statement. Participants either specified their favorite director/actor (music domain), music act (music domain) or author (book domain) and received recommendations based on the features of the works created by the artist (see Q9 in Listing 6.21, Sect. 6.8). The author applied a slightly different version of the query pattern depicted in Table 7.23 in each multimedia experiment. Therefore, the `<ENTITY-TYPE>` and `<PROPERTY>` parameters were set with domain-specific values (e.g., `dbo:Book` as `<ENTITY-TYPE>` and `dbo:author` as `<PROPERTY>` in the book domain) (Tab. 7.24). The engine generated recommendations based on the creative works of the favored artist. Participants also received suggestions from a simple *on-the-fly query*, which computed results according to the artist's characteristics (Tab. 7.23). In the experiment on music RS, recommendation lists comprised ten items, while for the studies on movie and book RS result sets only contained three items. It was assumed that in the latter domains participants would be able to assess the relevance of recommendations on a reduced set, such that task completion could be sped up. As in the travel experiment, recommendations from the *advanced rollup query* were compared with the *on-the-fly query* without letting participants know which approach they were assessing. Additionally, result set positions were randomly assigned on the screen.

After participants of the multimedia experiments had evaluated the results of TC3, they were guided to the *cross-domain* section (TC4), while the experiment on travel RS ended with TC3 because no suitable data could be identified to facilitate these kinds of suggestions. In the multimedia domain, however, the data from DBpedia was sufficient for this retrieval task. The entity type of the study (i.e., movie, music act or book) defined the source domain based on which the engine generated suggestions for items from another multimedia target domain. It was not possible to generate *cross-domain recommendations* for every potential entity type pair in the multimedia scenarios. For instance, no suitable graph patterns were identified in DBpedia to generate suggestions for music items based on preferences for books. However, for the remaining multimedia entity type pairs (i.e., movie/book, movie/music act), enough data sources existed to facilitate *cross-domain retrieval*. Participants formulated the corresponding queries by stating their favorite item from the source domain. Figure 7.27 depicts the *cross-domain* web form of the music experiment.

The web form assisted subjects in formulating a request, which then obtained either book or movie suggestions based on the favorite music act of the user. Parameters from users were entered in the placeholders of the query pattern in the backend of the application.

The second column of Table 7.25 shows two example *cross-domain queries* that tested the approach in the movie experiment. The other multimedia experiments used similar SKOSRec

Table 7.22: *On-the-fly* vs. *advanced rollup query* patterns in the travel experiment

Entity Type	Simple SKOSRec query	Advanced SKOSRec query
City	<pre>RECOMMEND ?city TOP 10 PREF dbr:<PREF_CITY> WHERE { VALUES ?cityType { yago:Town108665504 yago:City108524735 yago:UrbanArea108675967 dbo:City schema:City } ?city rdf:type ?cityType .}</pre>	<pre>SELECT ?city WHERE { VALUES cityType { yago:Town108665504 yago:City108524735 yago:UrbanArea108675967 dbo:City schema:City } ?location rdfs:subPropertyOf dul:hasLocation . ?poi ?location ?city . ?city rdf:type ?cityType . LIMIT 10 AGG dbr:<PREF_CITY> ?city SUM RECOMMEND ?poi TOP 1000 PREF dbr:<POI_1> dbr:<POI_2> dbr:<POI_3></pre>
Region	<pre>RECOMMEND ?region TOP 10 PREF dbr:<PREF_REGION> WHERE { VALUES ?regionType { yago:Region108630039 yago:Region108630985} ?region rdf:type ?regionType .}</pre>	<pre>SELECT ?region WHERE { VALUES ?regionType { yago:Region108630039 yago:Region108630985} ?location rdfs:subPropertyOf dul:hasLocation . ?poi ?location ?region . ?region rdf:type ?regionType . LIMIT 10 AGG dbr:<PREF_REGION> ?region SUM RECOMMEND ?poi TOP 1000 PREF dbr:<POI_1> dbr:<POI_2> dbr:<POI_3></pre>
Country	<pre>RECOMMEND ?country TOP 10 PREF dbr:<PREF_COUNTRY> WHERE { ?country dbo:Country .}</pre>	<pre>SELECT ?country WHERE { ?location rdfs:subPropertyOf dul:hasLocation . ?poi ?location ?country . ?country rdf:type ?countryType . LIMIT 10 AGG dbr:<PREF_COUNTRY> ?country SUM RECOMMEND ?poi TOP 1000 PREF dbr:<POI_1> dbr:<POI_2> dbr:<POI_3></pre>

Table 7.23: *On-the-fly* vs. *advanced rollup* query patterns in the multimedia experiments

Simple SKOSRec query	Advanced SKOSRec query
<pre>RECOMMEND ?recommendation TOP 3 PREF dbr:<PREF_ARTIST> WHERE { ?recommendation rdf:type <ENTITY_TYPE>}</pre>	<pre>SELECT ?recommendation WHERE { ?mediaItem <PROPERTY> ?recommendation. } LIMIT 3 AGG dbr:<PREF_ARTIST> ?recommendation SUM RECOMMEND ?mediaItem TOP 100 PREF [?prefItem WHERE { ?prefItem <PROPERTY> dbr:<PREF_ARTIST> }]</pre>

Table 7.24: Parameter for *advanced rollup* query patterns in the multimedia experiments

Exp.	Option	<ENTITY_TYPE>	<PROPERTY>
Movie	Director	yago:FilmDirector110088200	dbo:director
	Actor	yago:Actor109765278	dbo:starring
Music	N.A.	schema:MusicGroup	dbo:artist
Book	N.A.	dbo:author	dbo:author

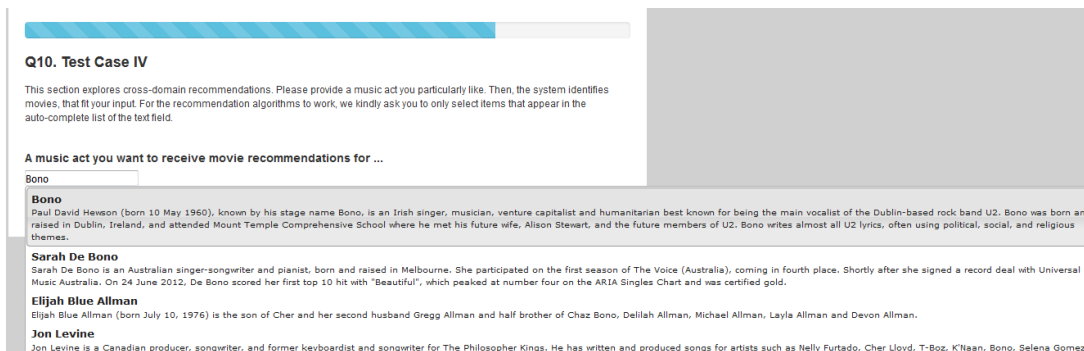


Fig. 7.27: Music - User profile generation, TC4 (page 9)

patterns which only switched entity types between subjects and objects of the triple. The author formulated the query templates in such a way that the query engine would match any properties or graph patterns that could potentially connect two items from the specified target and source domains. For instance, such matchings can occur when a user states his favorite movie that is written by a particular author. In case the same author has also written some books, they might be of interest to the user as well. Another example is when a consumer has declared a preference for a movie that links to the corresponding soundtrack. It might well be the case that the user likes the soundtrack and the music acts that were involved in creating it just as much as he/she likes the movie. The practice to distinguish homonymous LOD resources through the property `dbo:wikiPageDisambiguates` is another interesting linking pattern from DBpedia to be exploited for *cross-domain requests* since it connects similar entities (i.e., the book edition of a certain movie). Table 7.25 also lists SPARQL queries that were applied as the baseline

method in TC4. Both the SPARQL and SKOSRec queries are quite similar. However, while the SPARQL query retrieves target resources based on the connections to the favored LOD resource from the source domain, the SKOSRec query explores the same links for items that are similar to the preferred item. Hence, while the SPARQL query only matches *cross-domain* relations for a single LOD resource, the SKOSRec request expands the search space by considering links from more than one item. The *cross-domain* templates were formulated as aggregation-based query patterns. Unlike the previously presented *rollup queries* where sublevel entities are part of a larger entity, items in *cross-domain retrieval* are positioned at the same hierarchy level. Therefore, the maximum-based recommendation approach was picked for this task. Hence, the engine determined the final ranking score according to the most similar item from the source domain with connections to the target domain. As in the previous test cases, TC4 ended with an evaluation screen where users assessed the SPARQL-based as well as the SKOSRec suggestions in random order without knowing which recommendation list belonged to which method.

Table 7.25: SPARQL vs. SKOSRec query patterns for cross-domain recommendation retrieval

Op-tion	SPARQL Query	SKOSRec Query
Book	<pre> SELECT DISTINCT ?book WHERE { ?book rdf:type dbo:Book {?book dbo:basedOn dbr:<PREF_MOVIE> .} UNION { dbr:<PREF_MOVIE> dbo:basedOn ?book.} UNION { dbr:<PREF_MOVIE> dbo:genre ?genre. ?book dbo:literaryGenre ?genre.} UNION { dbr:<PREF_MOVIE> dbo:writer ?author. ?book dbo:author ?author.} UNION { ?dis dbo:wikiPage[...] dbr:<PREF_MOVIE> . ?dis dbo:wikiPage[...] ?book.} LIMIT 10 AGG dbr:<PREF_MOVIE> MAX RECOMMEND TOP 100 PREF dbr:<PREF_MOVIE> </pre>	<pre> SELECT ?book WHERE { ?book rdf:type dbo:Book {?book dbo:basedOn ?movie.} UNION { ?movie dbo:basedOn ?book.} UNION ?movie dbo:genre ?genre. ?book dbo:literaryGenre ?genre.} UNION { ?movie dbo:writer ?author. ?book dbo:author ?author.} UNION { ?dis dbo:wikiPage[...] ?movie. ?dis dbo:wikiPage[...] ?book.} LIMIT 10 AGG dbr:<PREF_MOVIE> MAX RECOMMEND TOP 100 PREF dbr:<PREF_MOVIE> </pre>
Music	<pre> SELECT DISTINCT ?musicGroup WHERE { ?musicGroup rdf:type schema:MusicGroup. { VALUES ?property { dbo:starring dbo:musicComposer dbo:associatedBand dbo:associatedMusicalArtist } { dbr:<PREF_MOVIE> ?property ?musicGroup.} UNION { dbr:<PREF_MOVIE> dbo:wikiPage[...] ?sound. ?sound dbo:genre dbr:Soundtrack. ?sound dbo:artist dbr:musicGroup. UNION { ?dis dbo:wikiPage[...] dbr:<PREF_MOVIE> . ?dis dbo:wikiPage[...] ?sound. ?sound dbo:genre dbr:Soundtrack. ?sound dbo:album ?soundtrack. ?sound dbo:musicalArtist ?musicgroup. } LIMIT 10 AGG dbr:<PREF_MOVIE> ?movie MAX RECOMMEND TOP 100 PREF dbr:<PREF_MOVIE> </pre>	<pre> SELECT ?musicGroup WHERE { ?musicGroup rdf:type schema:MusicGroup. { VALUES ?property { dbo:starring dbo:musicComposer dbo:associatedBand dbo:associatedMusicalArtist } { ?movie ?property ?musicGroup.} UNION { ?movie dbo:wikiPage[...] ?sound. ?sound dbo:genre dbr:Soundtrack. UNION { ?dis dbo:wikiPage[...] ?movie. ?dis dbo:wikiPage[...] ?sound. ?sound dbo:genre dbr:Soundtrack. ?song dbo:album ?sound. ?song dbo:musicalArtist ?musicgroup. } LIMIT 10 AGG dbr:<PREF_MOVIE> ?movie MAX RECOMMEND TOP 100 PREF dbr:<PREF_MOVIE> </pre>

Table 7.26: Response Rates for TC3

Domain	Regular	Rollup	N
Travel	99%	100%	103
Movie	96%	92%	50
Music	100%	100%	53
Book	96%	100%	51

7.6.2 Results of the Evaluation of Advanced Query Patterns

This subsection presents user assessments of *advanced recommendation requests*. At first, score values for TC3 are shown. In this test case, the SKOSRec engine almost always generated non-empty recommendation lists (see Tab. 7.26). In the travel and the book experiment, the response rate was higher for *rollup queries*, whereas in the movie domain, the engine more often provided recommendations for *regular requests*. However, these differences are only marginal and do not indicate a clear superiority of one method.

Fig. 7.28 depicts participants' general agreement to positive Likert statements concerning the *perceived usefulness* of the approaches in TC3. The diagram shows similar results for *on-the-fly* (i.e., regular) and *advanced requests* (i.e., *rollup queries*).

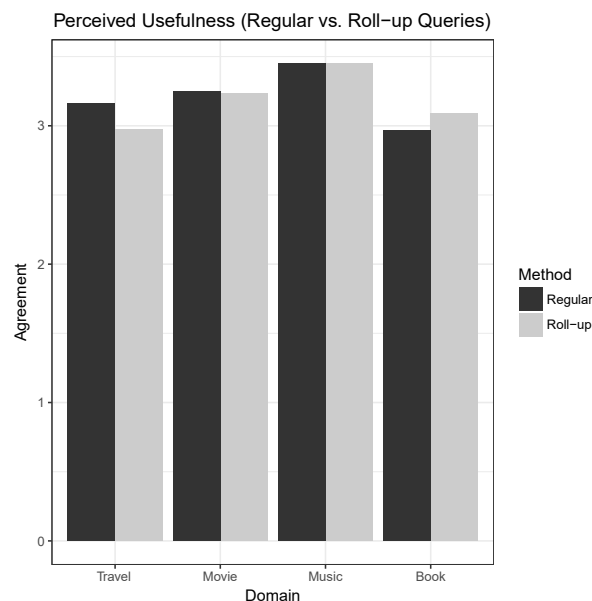


Fig. 7.28: Participants' overall satisfaction with *on-the-fly* (regular) and *rollup requests*

On average, satisfaction scores reached levels, which lay slightly above neutral agreement. A subsequent t-test confirmed no significant differences between the two methods. In fact, the approach with the best *usefulness* score was different in each experiment (see Tab. 7.27). While mean values were higher for regular requests in the studies on travel and music RS, *advanced queries* achieved better results in the movie and book experiments. Among the performance metrics, content-based diversity (*divC*) was the only one with systematic differences. Here,

scores went up for *advanced retrieval patterns* in the travel ($t(98) = -5.50, p < 0.001$), the music ($t(52) = -4.51, p < 0.001$) and the book domain ($t(47) = -4.90, p < 0.001$) (Tab. 7.27).

Table 7.27: Performance results of *on-the-fly* and *rollup recommendation* retrieval

Dimension (Metric)	Domain	Regular		Rollup		N
		M	SD	M	SD	
Accuracy (<i>size</i>)	Travel	<u>9.90</u>	0.99	9.42	2.07	103
	Movie	<u>2.88</u>	0.59	2.76	0.82	50
	Music	10.00	0.00	10.00	0.00	53
	Book	2.88	0.59	<u>3.00</u>	0.00	51
Accuracy (<i>mrs</i>)	Travel	<u>48.21</u>	24.98	44.93	25.59	102
	Movie	54.58	22.88	<u>56.69</u>	25.42	44
	Music	53.95	18.31	<u>58.24</u>	16.19	52
	Book	<u>51.53</u>	23.90	49.25	22.30	49
Novelty (<i>nv</i>)	Travel	<u>0.43</u>	0.39	0.38	0.40	98
	Movie	<u>0.35</u>	0.38	0.28	0.41	41
	Music	<u>0.49</u>	0.36	0.46	0.39	50
	Book	0.61	0.42	<u>0.62</u>	0.44	47
Diversity (<i>divU</i>)	Travel	<u>3.5</u>	-	3	-	100
	Movie	<u>4</u>	-	3	-	43
	Music	<u>4</u>	-	3	-	50
	Book	3	-	3	-	48
Diversity (<i>divC</i>)	Travel	0.79	0.17	0.89	0.13	99
	Movie	<u>0.75</u>	0.20	0.66	0.27	44
	Music	0.83	0.17	0.93	0.07	53
	Book	0.74	0.27	0.93	0.06	48
Usefulness	Travel	<u>3.36</u>	0.77	3.22	0.74	101
	Movie	3.20	0.92	<u>3.26</u>	0.94	44
	Music	<u>3.45</u>	0.84	3.44	0.85	51
	Book	2.95	0.87	<u>3.03</u>	0.96	48

Interestingly, users seemed to have perceived recommendations that were based on *on-the-fly requests* more diverse (*divU*) and new (*nv*) than suggestions resulting from *advanced queries*. Even though none of the mean/median scores in these performance dimensions was better according to t-tests, there may be minor effects that were not significant, due to the small samples. In the *novelty (nv)* and user-based *diversity (divU)* category, 3 out of 4 mean/median scores were higher. The results indicate an interesting contradiction: The *advanced retrieval pattern* was set up to improve the exploration of the knowledge graph. However, users' perceptions might have been slightly different. While the regular method generated unexpected suggestions, *advanced queries* produced recommendations that were more in line with the profile. Nevertheless, in spite of significantly improved content-based *diversity scores (divC)*, no conclusion can be drawn in favor of one method or the other, given the ambiguous results and the low to zero correlation between *divC* and *perceived usefulness*.

It is hypothesized that both approaches produce recommendation lists of similar quality and that they can provide benefits to potential consumers. It depends on the underlying information need, the respective domain and the available data in the LOD repository, whether or not a particular approach is better suited for a given retrieval task, since neither approach outperformed the other one. This conclusion is especially impressive, given the mean Jaccard scores for recommendation lists resulting from *on-the-fly* and *rollup retrieval* (see Tab. 7.28).

Table 7.28: JI of recommendation lists for TC3

Domain	Mean Jaccard Index (JI)	SD	N
Travel	0.07	0.11	101
Movie	0.03	0.07	44
Music	0.02	0.05	53
Book	0.02	0.06	50

The mean score never exceeded 0.1 throughout the domains which indicates a low concordance between result sets. It is remarkable that participants perceived the recommendations as equally good, given the high dissimilarity of the lists. Thus, *advanced queries* have an added-value, as they provide further relevant results. They can help to explore LOD repositories in different ways than a regular retrieval approach. Therefore, it is worthwhile to offer *rollup query patterns* as an alternative retrieval strategy when the regular approach has not produced any helpful recommendations.

User assessments of TC4 (*cross-domain recommendations*) were also evaluated. Table 7.29 lists the response rates for the SPARQL- and the SKOSRec-based *cross-domain requests*. The results show that the SKOSRec query had a higher success rate of generating non-zero result sets throughout the multimedia domains. Participants almost always received a recommendation for this query type, whereas in case a regular SPARQL query was executed, they often did not receive a single suggestion at all.

Table 7.29: Response rates for TC4

Domain	Regular (SPARQL)	Cross-Domain	N
Movie	32%	96%	50
Music	62%	98%	53
Book	53%	100%	51

Figure 7.29 gives an overview of participants' general satisfaction with *cross-domain recommendations* as opposed to regular suggestions resulting from a SPARQL request. The diagram indicates a tendency of participants to prefer recommendations from the SPARQL query. However, a Wilcoxon test did not confirm this assumption.

Additionally, the author applied non-parametric tests for the performance metrics *mrs*, *nv*, *divU*, *divC*, and *perceived usefulness*, due to the low response rates of the SPARQL-based approach, which led to a low number of comparable data points accordingly. Overall the

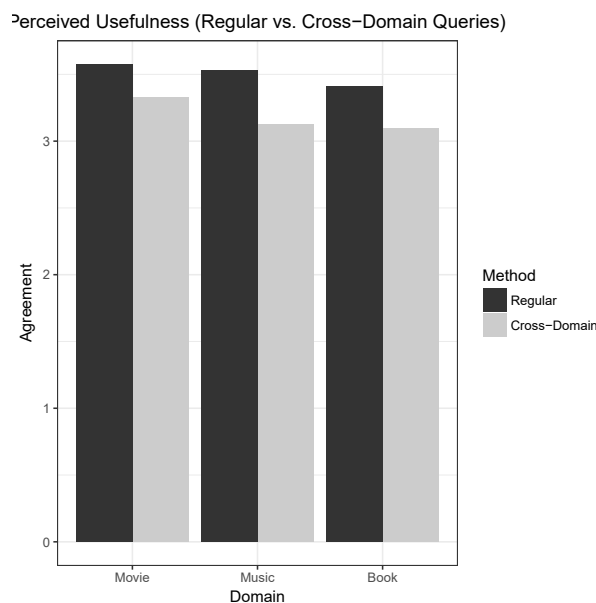


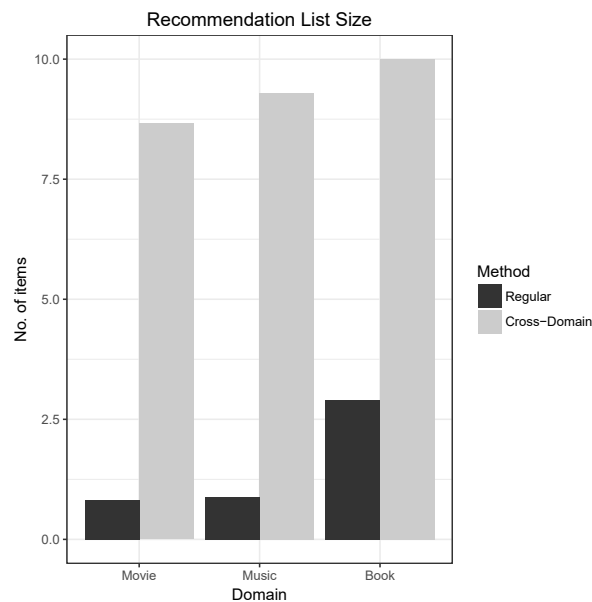
Fig. 7.29: Participants' overall satisfaction with *on-the-fly* (regular) and *cross-domain requests*

FDR-adjusted pairwise Wilcoxon tests detected no significant differences, except for user-based diversity scores ($divU$) in the movie ($Z = -2.69, p < 0.05$) and the book domain ($Z = -2.56, p < 0.05$). The results for the content-based diversity ($divC$) metric seem to point in the same direction because of the high mean values for *cross-domain queries* in each usage scenario. However, the significance of these results was not verified. *Precision* scores (mrs), on the other hand, indicate a better performance of regular SPARQL queries, but the within-subjects test did not confirm this assumption statistically (see Tab. 7.30).

However, the most important finding of the analysis is that *cross-domain queries* generated significantly larger lists throughout the domains. (movie: $t(49) = -17.92, p < 0.001$, music: $t(52) = -25.73, p < 0.001$, book: $t(50) = -12.75, p < 0.001$). Figure 7.30 illustrates this graphically. In the case of a successfully answered SPARQL query, the result set almost always contained only a few items due to data sparsity. In contrast to that, the SKOSRec engine was not only able to generate recommendations for practically every *cross-domain request*, it also mostly provided the required number of 10 recommendations. The verified increase in result set *size* is a remarkable outcome, given the fact that no significant differences were identified between the two approaches regarding *perceived usefulness*. Therefore, it is reasonable to assume that the capability of the SKOSRec engine to flexibly switch between processing steps of similarity calculation and graph pattern matching facilitates an improved exploration of LOD repositories. Table 7.31 also shows that often the items in the two sets often did not match, as is indicated by low Jaccard indices. This finding further demonstrates that the execution of an additional step of similarity calculation can help to retrieve other relevant items. Regarding practical implications, these results suggest that the two approaches could be combined into a single SKOSRec query pattern to achieve the best outcome.

Table 7.30: Performance results of SPARQL- and SKOSRec-based *cross-domain queries*

Dimension (Metric)	Domain	Regular (SPARQL)		Cross-Domain		N
		M	SD	M	SD	
Accuracy (<i>size</i>)	Movie	0.82	2.09	8.66	2.73	50
	Music	0.89	1.59	9.30	2.03	53
	Book	2.90	3.98	10.00	0.00	51
Accuracy (<i>mrs</i>)	Movie	<u>67.41</u>	27.62	53.65	14.71	16
	Music	<u>62.36</u>	31.76	48.32	22.49	20
	Book	<u>66.42</u>	21.28	55.59	16.84	27
Novelty (<i>nv</i>)	Movie	<u>0.59</u>	0.49	0.41	0.34	16
	Music	<u>0.72</u>	0.44	0.59	0.40	19
	Book	0.54	0.41	<u>0.63</u>	0.30	27
Diversity (<i>divU</i>)	Movie	3	-	4	-	16
	Music	3	-	3	-	20
	Book	3	-	4	-	27
Diversity (<i>divC</i>)	Movie	(0.64)	(0.35)	(0.99)	(0.02)	(6)
	Music	(0.90)	(0.13)	(0.98)	(0.03)	(10)
	Book	(0.84)	(0.12)	(0.87)	(0.1)	(8)
Usefulness	Movie	<u>3.57</u>	0.83	3.35	1.57	15
	Music	<u>3.53</u>	1.17	3.11	0.98	20
	Book	3.41	1.04	3.41	0.94	27

Fig. 7.30: Mean recommendation list *sizes* of SPARQL (regular) and SKOSRec *cross-domain queries*

In addition to performance comparisons between *cross-domain* and SPARQL requests, the author investigated how users perceived *cross-domain queries* in comparison to simple *on-the-fly recommendations*. For this purpose, the author conducted between-subjects t-tests comparing

Table 7.31: JI of recommendation lists for TC4

Domain	Mean Jaccard Index (JI)	SD	N
Movie	0.03	0.07	44
Music	0.01	0.02	21
Book	0.16	0.27	33

scores of the first test case (TC1) with evaluation data for *cross-domain queries* (TC4). Table 7.32 shows the outcome of this analysis.

Table 7.32: Performance results of *on-the-fly* (regular) and SKOSRec *cross-domain queries*

Dimension (Metric)	Domain	Regular			Cross-Domain		
		M	SD	N	M	SD	N
Accuracy (<i>size</i>)	Movie	10.00	0.00	50	8.66	2.73	50
	Music	10.00	0.00	53	9.30	2.03	53
	Book	9.80	1.40	51	<u>10.00</u>	0.00	51
Accuracy (<i>mrs</i>)	Movie	62.25	19.15	50	51.53	15.99	48
	Music	55.70	21.13	53	44.94	19.67	52
	Book	<u>57.47</u>	15.55	50	49.03	21.14	51
Novelty (<i>nv</i>)	Movie	0.36	0.31	50	0.54	0.36	43
	Music	0.36	0.39	52	0.58	0.42	50
	Book	0.61	0.29	47	<u>0.64</u>	0.32	51
Diversity (<i>divU</i>)	Movie	3	-	50	<u>4</u>	-	47
	Music	3	-	53	<u>4</u>	-	52
	Book	3	-	50	4	-	51
Diversity (<i>divC</i>)	Movie	0.84	0.12	50	0.98	0.03	47
	Music	0.86	0.10	53	0.99	0.02	51
	Book	0.92	0.08	50	0.93	0.10	23
Usefulness	Movie	<u>3.55</u>	0.73	50	3.33	1.11	48
	Music	<u>3.18</u>	0.85	53	3.13	0.94	52
	Book	<u>3.43</u>	0.81	50	3.10	1.11	51

Regarding *perceived usefulness*, the tests confirmed no significant differences, despite the increased mean scores for *on-the-fly queries* in each experiment. The *usefulness* scores for *cross-domain queries*, on the other hand, have a higher standard deviation, presumably because this kind of query pattern is dependent upon the data quality of the items and their interconnections. Nevertheless, these assumptions are only speculative and cannot be drawn conclusively from the user evaluations.

What is, however, indisputable is that both types of retrieval requests showed differences in the performance dimensions *accuracy*, *novelty* and *diversity*. Between-subjects t-tests revealed that *on-the-fly requests* achieved higher precision (*mrs*) and recall (*size*) scores, whereas *cross-domain queries* were more likely to produce surprising (*nv*) and topically diverse (*divC*) recom-

mentation lists.⁹ The significant differences occurred both in the movie and the music domain and are marked in bold figures accordingly (see Tab. 7.32). For the performance metrics of recommendation list *size*, (movie: $t(49) = 3.47, p < 0.01$, music: $t(52) = 2.5, p < 0.01$), *mrs* (movie: $t(96) = 3.00, p < 0.01$, music: $t(103) = 2.70, p < 0.01$) and *novelty* (movie: $t(91) = -2.49, p < 0.05$, music: $t(100) = -2.67, p < 0.05$) the effects were fairly small and quite large for *divC* (movie: $t(55) = -8.05, p < 0.001$, music: $t(55) = -8.87, p < 0.001$) scores.

In summary, it can be concluded that *cross-domain queries* can provide more surprising and diverse recommendations than regular *on-the-fly requests* and might therefore enhance existing content-based strategies with an additional interesting retrieval approach.

⁹A non-parametric Mann-Whitney test was applied to detect differences in user-based diversity scores (*divU*).

8 Conclusion

This chapter presents the main findings of the dissertation project. Section 8.1 summarizes the capabilities of the SKOSRecommender as well as the primary results of the requirement-specific evaluation. Section 8.2 discusses the limitations of the chosen experimental methods, outlines future research directions and provides practical recommendations for real-world applications.

8.1 Summary

The assumption that LOD resources are currently insufficiently used by existing recommender systems is at the core of this thesis. Starting with the characteristic features of RDF data, the author identified the strengths and challenges of these data sources for content-based RS. The LOD cloud contains many collections of public interest, which often hold extensive and timely information on numerous real-world objects. On the other hand, LOD usage has many challenges. For instance, data sources and data models are heterogeneous, since the data web contains many different vocabularies. Other problems concern the quantity and distribution of the data. Based on the characteristics of the LOD cloud, a requirements specification for a LOD-enabled recommendation engine was defined. The purpose of the specification was to comprehensively reflect the strengths and weaknesses of RDF data for recommendation tasks. A subsequent literature survey compared these specifications with the features of existing non-Linked Data and Linked Data retrieval engines. The review showed that there does not exist an RS or IR system that thoroughly reflects the characteristics of the LOD cloud.

The SKOSRecommender engine, which was developed in the course of this thesis, implements the required system features. It integrates existing CB recommendation techniques with query-based retrieval to improve usage of LOD resources for personalized search. In this context, the capability of *on-the-fly* processing plays a significant role, since it enables flexible combinations of graph-based query parts with IR methods. With this feature, the engine can feed results from previous query stages into subsequent processing steps of the recommendation workflow. By these means, it becomes possible to identify existing semantic connections and graph patterns as well as implicit information from LOD repositories.

An optimized approach of *on-the-fly retrieval* was proposed and tested to handle the weaknesses of LOD processing regarding data quantity accordingly. Another feature that addresses a challenge of the LOD cloud is the usage of SKOS annotations and vocabularies for similarity calculation. Since many public data repositories apply SKOS vocabularies, a generic approach

to similarity calculation can make use of these annotations for various LOD datasets thereby at least partially circumventing the problem of data heterogeneity. Many SKOS vocabularies state mapping relations to other thesauri on the LOD cloud. They can be taken advantage of to generate *cross-repository recommendations*. Thus, the application of SKOS vocabularies addresses the issue of data distribution of the LOD cloud as well.

While being primarily applied to handle weaknesses of LOD retrieval, interlinked knowledge organization systems can be utilized to facilitate flexible adaptations of similarity calculation by incorporating semantically related concepts into the retrieval process. The *flexible similarity detection* method of this thesis extends existing approaches of concept expansion by taking inter-concept similarities into account.

In addition to features that primarily address potentials and challenges of LOD-enabled recommendations, the SKOSRec system works well with the LOD technology stack. For instance, the engine provides SPARQL-like query options and an interface to SPARQL endpoints. The ability of the system to issue SPARQL-like queries has been realized by the definition of the SKOSRec query syntax and the implementation of a corresponding SKOSRec parser and compiler. By using the SKOSRec query language, it is possible to apply, adapt and combine the developed system features of *flexible similarity detection* and *on-the-fly, constraint-based* or *cross-repository retrieval* thus facilitating new and *advanced recommendation requests*. The query syntax contains graph-based elements as well as parts that trigger the calculation of item similarities. The additional advantage of the SKOSRec query processor is that it can seamlessly retrieve data from SPARQL endpoints over HTTP requests, which allows obtaining recommendations from remote LOD repositories. The prerequisite for this is that data collections contain SKOS annotations.

In addition to the successful prototypical implementation of system features from the requirements specification, it was also important to investigate whether the new query options provide added value to conventional content-based methods of LOD retrieval. After all, the central hypothesis of this work claims that current LDRS have so far made too little use of the strengths of the LOD cloud for recommendation tasks. For this purpose, the author carried out evaluations for representative usage scenarios (i.e., DL, travel, movie, music and book RS) and LOD repositories (i.e., EconStor and DBpedia) to determine whether SKOSRec suggestions can compete with, if not outperform, recommendations from conventional methods.

The quantitative performance was measured by throughput rates and response times. Offline simulation runs showed that the optimized retrieval approach significantly reduced the workload (i.e., the number of records that need to be processed) thus achieving scalability and linear growth of response times for larger record sizes. In the regular execution mode of *on-the-fly retrieval*, similarity calculation had an average duration of a few seconds (or a few hundreds of milliseconds for smaller datasets), while the approach of *fast on-the-fly retrieval* considerably reduced processing periods for high throughput rates. On the other hand, with small data collections or low throughput rates (e.g., in the domain of DL and travel destination search),

response times were slightly increased. In these cases, the approach caused too much overhead. Hence, it should only be applied when the system accesses large collections and therefore needs to provide sufficient scalability.

In addition to quantitative performance, recommendation quality was evaluated in online and offline experiments. Common metrics assessed RS quality regarding algorithmic performance. In particular, the following quality dimensions were measured: *accuracy*, *novelty* and *diversity*. The online experiments also gathered user opinions on the *perceived usefulness* of suggestions. By carrying out experiments over several datasets measuring different dimensions, it was intended to capture the engine's actual performance for the various query types and retrieval methods as accurately as possible.

In this context, the experiments tested whether the SKOSRec engine in its basic configuration of content-based *on-the-fly retrieval* is capable of generating suggestions of reasonable quality. Since this method closely resembles approaches of existing LDRS, the user evaluations for these recommendations served as baseline data. The assessments of the baseline method were mostly positive. In the DBpedia domains, on average, the majority of users confirmed that the engine provides useful suggestions. Evaluations in the DL domain, however, pointed in the opposite direction. Here, users tended to reject positive statements regarding baseline recommendations. In addition to the aspect of general recommendation quality, the online experiments also evaluated the influence of the metrics in the quality dimensions of *accuracy*, *novelty* and *diversity* on the *perceived usefulness* construct. The results confirmed findings from previous research, which emphasize *accuracy* as the most important predictor of overall user satisfaction. The remaining quality aspects (*diversity* and *novelty*), though not entirely irrelevant, are of secondary importance. They only had a marginal impact on *perceived usefulness*. The weighting of the individual metrics influenced the interpretation of subsequent evaluation results concerning more advanced retrieval techniques.

For instance, it was applied to assess the performance of the *flexible similarity detection* method, which was mostly tested in offline experiments. In the DBpedia domains, the simulation runs revealed that the *flexible similarity detection* method significantly improves *accuracy*, while still achieving good *diversity* and *novelty* scores. At the same time, the two methods used for comparison (concept expansion on `skos:broader` links and exact concept matching) either showed significant strengths in terms of *accuracy* with strong declines in *novelty* and *diversity* (exact concept matching) or had good *novelty* and *diversity* scores, but performed poorly in terms of *accuracy*. It can be assumed that, on average, the *flexible similarity detection* approach achieves better results than a concept expansion on `skos:broader` links, since the unweighted consideration of similar SKOS concepts leads to noisy data, due to non-normalization of annotations. In this context, however, it is also important to note that the approach was not equally successful in the domain of DL search. Hence, it depends on the domain, the applied SKOS vocabulary and an adequate parameter tuning whether the method works. Once a suitable configuration is found, the *flexible similarity detection* approach is a veritable retrieval option.

Subsequent online experiments tested the approaches of *constraint-based* and *advanced recommendation queries*. One question was whether filtering improves the quality of suggestions. While a faceted search option is a common practice in IR systems, it is not as widespread in RS. Therefore, it was interesting to find out, if users perceived it as helpful. The majority of participants commented positively on this option throughout the usage scenarios. Therefore, it can be assumed that the approach of *prefiltered constraint-based* retrieval is a viable alternative recommendation strategy when previous attempts have yielded too many irrelevant items. However, it was even more important to find out to what extent the graph structures of the LOD cloud can be used for improved *constraint-based retrieval*. While simple attribute-level filters can be realized with a flat table structure, *expressive query* patterns require semantic network information. For this purpose, the author formulated graph-based filter patterns for each DBpedia-based usage scenario. These filter patterns were assumed to take good advantage of the available knowledge sources. It could be shown that an *expressive user filter* increases response rates. Throughout the domains, the quota of responses with a non-empty result set compared to all responses was higher when an *expressive filter* was applied. This difference was most evident in the travel domain. Additionally, result set *sizes* (which are assumed indicators of *recall* in the *accuracy* dimension) were significantly higher with *expressive filter* options. For all other performance metrics (e.g., *precision*, *novelty* or *diversity*) these differences were not quite as clear. Although for some of the remaining metrics, mean scores were slightly increased in the *expressive filtering* mode, these differences were not significant. Hence, an overall superiority of expanded constraints cannot be ascribed. However, it can be assumed that *expressive filtering* achieves a competitive level of quality in the remaining performance dimensions. In summary, it can be stated that graph-based query constraints might help to increase response rates and recommendation list *sizes*, while still providing the same level of quality as *simple filters*.

The same result of improved *recall* values was achieved for *cross-domain queries*. These kinds of requests are examples for *advanced retrieval patterns* that combine graph pattern matching with similarity calculation. In all three multimedia domains, both the response rates and the *recall* values were significantly increased when a SKOSRec *cross-domain* instead of a regular SPARQL request was issued. Likewise, for two domains, the user-based *diversity* ratings showed a significant improvement. This is a result that might be at least partly attributed to the enlargement of recommendation lists. For all other metrics, there were neither significant improvements nor declines in performance. Hence, the engine's novel recommendation approach might help to increase *recall* values while maintaining the same level of quality when applied in the context of *cross-domain queries*.

These findings suggest that the fusion of item similarity computation and graph-based retrieval is indeed compelling and can help to optimize exploitation of existing knowledge sources from the LOD cloud.

It was also evaluated whether SKOSRec *cross-domain recommendations* can compete with suggestions that were generated through regular *on-the-fly retrieval*. The statistical tests revealed

that the regular approach provides more accurate recommendations, whereas *cross-domain requests* can help to surprise users. Once users are interested in receiving suggestions from a different domain than their preferred items, they will be presented with diverse recommendation lists containing more novel items than a regular solution set resulting from *on-the-fly retrieval*. For the group of *advanced queries*, it was also evaluated, whether *rollup* patterns can improve results. There were only marginal variations between quality levels of *on-the-fly* and *rollup recommendations*. Significant differences were only found for content-based *diversity* scores (*divC*), which favored *rollup* over regular requests. However, when interpreting the results, one has to be careful to conclude that this query type improves quality because baseline analyses had not found any correlations between *divC* scores and *perceived usefulness*. Additionally, user-based *diversity* scores (*divU*) were not significantly higher for the *advanced queries*. Although *rollup* patterns generated diversified recommendation lists, they did not change user perceptions in this regard. Moreover, in all other performance dimensions, no significant differences were identified either. Throughout the domains and except for *divC* scores, sometimes the one and sometimes the other approach achieved higher mean scores. Since these fluctuations were not significant, it might depend on the specific recommendation request whether one method is better than the other. Give these findings, it is hypothesized that *rollup* patterns are an interesting alternative retrieval strategy.

In summary, it can be stated that the SKOSRec engine addresses the challenges of LOD processing while at the same time taking advantage of the characteristics of LOD knowledge sources. Despite the fact that the SKOSRec system successfully implements the specified system features, offline and online experiments have revealed that new recommendation approaches mostly fulfill the qualitative criteria of the requirements specification. The evaluations have shown that the SKOSRec system can often generate relevant and useful suggestions when novel recommendation approaches are applied. In many cases, the application of *expressive* or *advanced query patterns* (i.e., in the context of *constraint-based* or *cross-domain retrieval*) helps to improve *recall* values, while still providing the same level of quality in the remaining performance dimensions. *Advanced requests* can be used to generate new or diversified recommendation lists in case users are not satisfied with the results of regular requests (i.e., *rollup* or *cross-domain* vs. *on-the-fly queries*). The synopsis in Table 8.1 relates each requirement to the respective system feature of the SKOSRec prototype to demonstrate the system's overall capabilities.

From this summary, it can be seen that the system meets almost all of the objectives. Qualitative features that only partly meet the requirements specification (i.e., where superiority over regular LOD retrieval did not reach significant levels across the majority of domains) are shown in parentheses. In these cases, it depends on the usage scenario, whether the novel approach can provide improved result lists. Despite the fact that no general statements are possible for these requirements, the new recommendation approaches extend existing methods and at least provide the option of an alternative search strategy when conventional methods have failed.

Table 8.1: Functional & Qualitative System Features

Requirement	Fulfillment	Features of the SKOSRec system
RQ1	✓	On-the-fly similarity calculation from LOD repositories.
RQ2	✓	Similarity calculation through SKOS vocabularies and DCMI annotations.
RQ3	✓	Flexible similarity calculation through SKOS-enabled concept expansion and precomputation of concept-to-concept similarity values.
RQ4	✓	Integration of (remote) SPARQL endpoints.
RQ5	✓	SKOSRec query language (incl. SPARQL syntax elements and language processing units).
RQ6	✓	Ability of the SKOSRec query language to formulate advanced recommendation requests.
RQ7	✓	Ability of the SKOSRec query language to formulate cross-repository recommendation requests.
RQ8	✓	Ability of the recommendation engine to process cross-repository recommendation requests.
RQ9	✓	Ability of the recommendation engine to provide quick responses to single-item on-the-fly recommendation requests.
RQ10	✓	Ability of the recommendation engine to handle large datasets well.
RQ11	✓	Ability of the recommendation engine to improve recommendation quality through flexible similarity detection methods.
RQ12	✓	Ability of the recommendation engine to improve recommendation quality through constraints.
RQ13	(✓)	Ability of the recommendation engine to improve constraint-based recommendation retrieval through expressive SPARQL-like graph constraints.
RQ14	(✓)	Ability of the recommendation engine to improve recommendation quality through application of advanced query patterns.

8.2 Discussion

In addition to the positive outcome of the research project, the limitations of the results also need to be acknowledged. This refers to the generalizability of the study findings as well as to the technical restrictions of the SKOSRec system. Regarding the validity of the results from the experiments, practitioners should consider the following aspects: Even though the evaluation strictly followed established standards from RS research - due to limited time and financial resources - some trade-offs had to be made. For instance, it is arguable whether the number of usage scenarios can provide a representative overview of the range of typical recommenda-

tion tasks as well as over existing domains in the LOD cloud. Naturally, information needs are much more diverse than the examined recommendation tasks of this thesis. In theory, there exists a wide variety of potential query patterns and numerous additional usage scenarios, based on which the SKOSRec engine could have been tested. The selection of suitable application scenarios focused on the prevalence of common recommendation domains from industry and practice as well as on the existence of corresponding and suitable LOD collections. Future research will have to examine the effectiveness of the presented retrieval approaches for other domains and datasets accordingly. It also needs to be taken into account that the author did not test the novel recommendation strategies in each specified usage scenario. Due to missing graph patterns or time restrictions, some methods (e.g., *flexible similarity detection* or *advanced query patterns*) could not be evaluated in every domain or experiment.

During the online studies, it would have also been desirable to let users explain their opinions on recommendation quality in greater detail. For instance, a free comment field below result lists might have grasped users' perceptions more comprehensively. Additionally, reversed statements could have increased the reliability of the *perceived usefulness* construct. However, due to the assumed limited attention span of participants, these items were omitted. Reasonable reliability levels suggest that this was a justified decision.

Aside from the limitations with regard to the evaluation, it is a strength of the developed approaches that they facilitate combinations of graph-based and similarity-based retrieval at different stages of the recommendation workflow. This feature extends general search capabilities for semantic networks. It requires future research to investigate whether, in addition to SKOS, further RDF vocabularies can be used for item-to-item similarity computation to increase the range of potential usage scenarios as well as to extend recommendation approaches to be applicable to general search tasks. In this context, it also needs to be determined whether the representation of the user profile in the SKOSRec query language can be substituted with free-text expressions to enable search-like functionalities. LOD researchers have already developed engines that can process natural language queries over RDF data [143, 241, 263]. It will have to be investigated how the SKOSRec engine profitably fits into this landscape of existing NLP tools for RDF retrieval.

Another open research question concerns the aspect of interfaces to other applications. In its standard configuration, the system is not an end-user retrieval engine, but a backend application with which an administrator who has domain knowledge (e.g. of RDF vocabularies and data models of a particular usage scenario) creates the respective query patterns. Although the websites of the online experiments represent possible implementations of suitable end-user applications for the SKOSRec engine, there are still many other possibilities and extensions to design such a user interface (UI). For instance, the selection of LOD items for the profile was made possible by storing the data in an index that could then be accessed and searched before issuing a recommendation query. However, this runs against the idea of ad-hoc retrieval. Future research will have to determine whether it is feasible to generate an on-the-fly mapping from

items (e.g., in the product catalog of a commercial application) to the respective LOD resources. Even if it is not yet clear which UI components can make the best possible use of the SKOSRec engine's features, the results from the user experiments can give at least a few suggestions. The following retrieval options might be the default setting in an interface that contains a LOD-enabled RS component to assist users in finding interesting items. Since the regular approach of *on-the-fly retrieval* achieved good results throughout the domains, the display of simple recommendations generated from previously stated user preferences represents a suitable starting point for retrieval. *On-the-fly queries* could be refined by similarity thresholds that lead to the incorporation of semantically related SKOS concepts (i.e., concept expansion through *flexible similarity calculation*) and a re-ranking of suggested items in case a user makes a corresponding statement (e.g., through a slider option). Additionally, there should be a button to filter recommendation results. Since the evaluations have shown that expressive constraints often lead to increased recall values, the application of such a filter might be the best option for assisted retrieval. As in the preparations for the experiments, appropriate graph-based query patterns for the usage scenario in question would have to be determined by a domain expert before setting up the application. The same applies to *advanced retrieval patterns*, which should be available to the end user to refine the query when both the regular as well as the *constraint-based approach* have failed to provide relevant items. This suggestion is made because *rollup queries* were competitive with regular recommendations in the web-based experiments. Thus, they represent a viable alternative retrieval strategy. For some domains, it is also possible to use an optional selection field to enable the formulation of *cross-domain requests*. Evaluations for this query type suggest that users might receive interesting results (i.e., unexpected items or diversified recommendation lists) from such a request.

The prototypical implementation and the evaluation of the SKOSRec engine in different usage scenarios have demonstrated that the freely available knowledge sources of the LOD cloud can be successfully applied to advance LOD-enabled retrieval approaches. The novel SKOS-based recommendation strategies proposed by this work, namely *fast on-the-fly retrieval*, *flexible similarity detection*, *cross-repository recommendations*, *(expressive) graph-based filters* and *advanced recommendation queries* (i.e., *rollup* and *cross-domain requests*) are promising alternatives to existing IR technologies.

A Code Examples of the SKOS Recommender

A.1 Configuration and Query Input

```

package rec;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import rec.ParserRec;
import rec.RecCompiler;
import rec.RecQuery;
import federation.CollectionContainer;
import federation.Configuration;
import federation.DistributedResourceCollection;
import federation.ResourceCollection;
import federation.SKOSMapping;
import federation.SKOSVocabulary;
import federation.Service;

public class Mainer {
    public static void main(String[] args) throws IOException, InterruptedException {
        // Service object to access the SPARQL API of the default repository (HTTP address, maximum no. of records)
        Service econstor = new Service("http://localhost:8890/sparql", 10000);
        // Vocabulary object to access SPARQL API of the SKOS vocabulary
        SKOSVocabulary cat = new SKOSVocabulary(econstor);
        // Collection object representing the default repository
        (service object, annotation property of the default repository, vocabulary object of the target collection)
        ResourceCollection agris = new ResourceCollection(econstor, "http://purl.org/dc/terms/subject", cat);
        // Distributed collection (default repository, annotation property of the default repository, vocabulary object)
        DistributedResourceCollection distributed = new DistributedResourceCollection(econstor, "http://purl.org/dc/elements/1.1/subject", cat);
        // Sets the target collection of the distributed repository
        distributed.setTargetCollection(new CollectionContainer(agris));
        /*
         * Sets the vocabulary objects to access the SKOS vocabularies of the distributed collection
         * Only a single object is needed to access the two vocabularies STW and AGROVOC in this example
         */
        List<SKOSVocabulary> skosList = new ArrayList<SKOSVocabulary>();
        skosList.add(cat);
        // Sets the SKOS mapping object
        distributed.setSKOSMapping(new SKOSMapping(cat, skosList));
        // Sets the configuration for the distributed retrieval process
        Configuration conf = new Configuration(new CollectionContainer(distributed), true);
        // Sets the SKOSRec query string
        String queryStr = "PREFIX econ: <http://linkeddata.econstor.eu/beta/resource/publications/> "
            + "PREFIX econColl: <http://linkeddata.econstor.eu/beta/resource/collection/>"
            + "RECOMMEND ?item TOP 10 "
            + "PREF econ:21555 SIM > 0.75 ";
        // SKOSRec query object
        RecQuery recquery = new RecQuery();
        // SKOSRec parser object
        ParserRec parser = new ParserRec();
        // Parses the query string with the SKOSRec parser to verify syntactic correctness
        RecQuery parsedQuery = parser.parseRec$(recquery, queryStr);
        // SKOSRec compiler object
        RecCompiler compiler = null;
        // Solution object
        List<Map<String, Object>> res = null;
        // Compiles the SKOSRec query and puts out the solution upon successful compilation
        try {
            compiler = new RecCompiler();
            res = compiler.compile(parsedQuery, conf);
        }
        catch (Exception ex) {
            res = null;
        }
    }
}

```

```

    if ( res != null ) {
        System.out.println("Ergebnis: "+res);
    }
}

```

A.2 SKOSRec Grammar

```

/**
 * Licensed to the Apache Software Foundation (ASF) under one
 * or more contributor license agreements. See the NOTICE file
 * distributed with this work for additional information
 * regarding copyright ownership.
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * This grammar file is a derivative version of the .jj file for
 * SPARQL 1.1 of Apache Jena ARQ that can be obtained from
 *
 *      https://github.com/apache/jena/blob/master/jena-arq/Grammar/sparql_11.jj
 *
 * It was modified to meet the demands of the SKOSRec grammar.
 */
[...]
void Query() : {Element where; }
{
    Prologue() ( SelectPart() )? SimProjection() (ItemPart()+ ( where = RecWhereClause())?
    {
        if (where != null) getQuery().setWhereClause(where.toString());
    }
}
void Prologue() : { }
{
    ( BaseDecl() | PrefixDecl() )*
}
void BaseDecl() : { String iri ; }
{
    <BASE> iri = IRIREF()
    { getPrologue().setBaseURI(iri) ; }
}
void PrefixDecl() : { Token t ; String iri ; }
{
    <PREFIX> t = <PNAME_NS> iri = IRIREF()
    { String s = fixupPrefix(t.image, t.beginLine, t.beginColumn) ;
      getPrologue().setPrefix(s, iri) ;
    }
}
// ---- Query type clauses
void SelectPart() : { }
{
    SelectQuery() (Aggregation())?
}

void SelectQuery() : { Element el;}
{
    SelectClause()
    ( DatasetClause() )*
    el = RecWhereClause() { getQuery().getSelectQuery().setQueryPattern(el) ; }
    (LimitClause())?
}

void SelectClause() : { Var v ; Expr expr ; Node n ; }
{
    <SELECT>
    { getQuery().getSelectQuery().setQuerySelectType() ; }
    ( <DISTINCT> { getQuery().getSelectQuery().setDistinct(true); }
    | <REDUCED> { getQuery().getSelectQuery().setReduced(true); } )?
    { allowAggregatesInExpressions = false ; }
    (
        (
            v = Var() { getQuery().getSelectQuery().addResultVar(v) ; }
        )+
    |
    <STAR> { getQuery().getSelectQuery().setQueryResultStar(true) ; }
    )
    { allowAggregatesInExpressions = false ; }
}

```

```

}

void Aggregation () : {String aggr; String v; Token type;}
{
    <AGG> aggr = iri() v = Var() ( type = <SUM> | type = <MAX> | type = <AVG> ) (ServiceIntegration())?
    { getQuery().setAggregation(type.image,aggr,v); }
}

String ServiceIntegration () : {String acc;}
{
    <FROM_SERVICE> acc = IRIREF()
    { return acc; }
}

void SimProjection() : {String n ; Token t; String service = ""; }
{
    <RECOMMEND> n = Var() <TOP> t = <INTEGER> (ServiceIntegration())?
    {if (service == "")
    getQuery().setSimProjection(n,integerValue(t.image));
    else
    getQuery().setSimProjectionService(n,integerValue(t.image),service);
    }
}

void ItemPart () : {Token dec = null; VarPart var = null; String iri; SimCondition sim = null;}
{
    <PREF> (dec = <DECIMAL>)? (var = VarPart() | iri = iri()) ( sim = Sim())?
    {if (var == null)
    if (sim == null)
    if (dec == null)
    getQuery().addItemPart(iri);
    else
    getQuery().addItemPart(doubleValue(dec.image),iri);
    else
    if (dec == null)
    getQuery().addItemPart(iri,sim);
    else
    getQuery().addItemPart(doubleValue(dec.image),iri,sim);
    else
    if (sim == null)
    if (dec == null)
    getQuery().addVarPart(var);
    else
    getQuery().addVarPart(doubleValue(dec.image),var);
    else
    if (dec == null)
    getQuery().addVarPart(var,sim);
    else
    getQuery().addVarPart(doubleValue(dec.image),var,sim);
    }
}

VarPart VarPart() : {String var; Element where;}
{
    <LBRACKET> var = Var() where = RecWhereClause() <RBRACKET>
    { return new VarPart(var.toString(), where.toString()); }
}

SimCondition Sim() : {String rel; Token t;}
{
    <SIM> rel = Relation() t = <DECIMAL>
    { return new SimCondition(rel,doubleValue(t.image)); }
}

String Relation() : {Token t ;}
{
    ( t = <EQ> | t = <GT> | t = <GE> )
    {return t.image ;}
}

void DatasetClause () : {}
{
    <FROM>
    ( DefaultGraphClause() | NamedGraphClause() )
}

```

```

void DefaultGraphClause() : { String iri ; }
{
  iri = SourceSelector()
  {
    // This checks for duplicates
    getQuery().getSelectQuery().addGraphURI(iri) ;
  }
}

void NamedGraphClause() : { String iri ; }
{
  <NAMED>
  iri = SourceSelector()
  {
    // This checks for duplicates
    getQuery().getSelectQuery().addNamedGraphURI(iri) ;
  }
}

String SourceSelector() : { String iri ; }
{
  iri = iri() { return iri ; }
}

Element RecWhereClause() : { Element el ; }
{
  (<WHERE>)?
  { startWherePattern() ; }
  el = RecGroupGraphPattern() { return el ; }
  { finishWherePattern() ; }
}
[...]
Element RecGroupGraphPattern() : { Element el = null ; Token t ; }
{
  t = <LBRACE>
  { int beginLine = t.beginLine; int beginColumn = t.beginColumn; t = null; }
  (
    { startSubSelect(beginLine, beginColumn) ; }
    {
      Query q = endSubSelect(beginLine, beginColumn) ;
      el = new ElementSubQuery(q) ;
    }
  | el = RecGroupGraphPatternSub()
  )
  <RBRACE>
  { return el ; }
}
[...]
Element RecGroupGraphPatternSub() : { Element el = null ; }
{
  { ElementGroup elg = new ElementGroup() ; }
  { startGroup(elg) ; }
  (
    { startTriplesBlock() ; }
    el = TriplesBlock(null)
    { endTriplesBlock() ; }
    elg.addElement(el) ; }
  )?
  (
    el = GraphPatternNotTriples()
    { elg.addElement(el) ; }
    (<DOT>)?
    (
      { startTriplesBlock() ; }
      el = TriplesBlock(null)
      { endTriplesBlock() ; }
      elg.addElement(el) ; }
    )?
  )*
  { endGroup(elg) ; }
  { return elg ; }
}
[...]
Element RecGraphPatternNotTriples() : { Element el = null ; }
{
  (
    el = RecGroupOrUnionGraphPattern()
  |
  el = RecOptionalGraphPattern()

```

```

|
| el = RecMinusGraphPattern()
|
| el = Filter()
|
| el = Bind()
|
| el = InlineData()
|
| { return el ; }
}

Element RecOptionalGraphPattern() : { Element el ; }
{ <OPTIONAL> el = RecGroupGraphPattern()
  { return new ElementOptional(el) ; }
}
[...]
Element RecMinusGraphPattern() : { Element el ; }
{
  <MINUS_P>
  el = RecGroupGraphPattern()
  { return new ElementMinus(el) ; }
}

Element RecGroupOrUnionGraphPattern() :
{ Element el = null ; ElementUnion el2 = null ; }
{
  el = RecGroupGraphPattern()
  ( <UNION>
  { if ( el2 == null )
    {
      el2 = new ElementUnion() ;
      el2.addElement(el) ;
    }
  }
  el = RecGroupGraphPattern()
  { el2.addElement(el) ; }
  )*
  { return (el2==null)? el : el2 ; }
}

```

A.3 SKOSRec Query

```

package rec;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import org.apache.jena.iri.IRIFactory;
import com.hp.hpl.jena.query.Query;
import com.hp.hpl.jena.sparql.core.Prologue;
import federation.Configuration;

public class RecQuery extends Query {
  private Configuration conf;
  private Query selectQuery;
  private Aggregation aggregation;
  private String whereClause;
  static IRIFactory iriFactory = IRIFactory.semanticWebImplementation();
  private SimProjection simprojection;
  private List<ItemPart> itemPartList = new ArrayList<ItemPart>();
  protected Prologue prologue;
  // @param config the configuration object of the running SKOSRec instance
  public RecQuery(Configuration config) {
    super();
    this.conf = config;
    selectQuery = new Query();
  }
  // Sets a new postfilter SELECT query
  public void setSelectQuery () {
    selectQuery = new Query();
  }
  // @return Determines whether the postfilter SELECT query is empty
  public boolean hasSelectQuery () {
    if (!selectQuery.toString().equals("")){ return true; }
    else { return false; }
  }
}

```

```

// Returns the postfilter SELECT query
public Query getSelectQuery() {return selectQuery;}
/*
 * Sets a prefilter condition for the SKOSRec query
 * @param where the prefilter condition
 */
public void setWhereClause(String where) {
    if (where == null) {this.whereClause = null;}
    else {this.whereClause = where.toString();}
}
// Determines whether the SKOSRec query contains a prefilter condition
public boolean hasWhereClause() {
    if (this.whereClause != null){ return true;}
    else { return false; }
}
// Returns the prefilter condition
public String getWhereClause() {
    return this.whereClause;
}
/*
 * Sets the parameters needed for recommendation retrieval
 * @param v the recommendation variable
 * @param l the number of recommendations to be returned
 */
public void setSimProjection (String v, int l) throws Exception {
    test(v);
    this.simprojection = new SimProjection(v,l);
}
/*
 * Sets the parameters needed for recommendation retrieval
 * @param v the recommendation variable
 * @param l the number of recommendations to be returned
 * @param s the SPARQL endpoint address for cross-repository retrieval
 */
public void setSimProjectionService (String v, int l, String s) throws Exception {
    test(v);
    this.simprojection = new SimProjection(v,l,s);
}
/*
 * Returns the parameters needed for recommendation retrieval
 * @return simprojection the SimProjection object of the SKOSRec query
 */
public SimProjection getSimProjection () { return this.simprojection; }
// Tests whether the recommendation variable is contained in potential pre- and postfilter conditions
private void test(String v) throws Exception {
    if (hasWhereClause()) {
        if (!this.whereClause.contains(v)) {
            throw new Exception("Recommendation variable not contained in WhereClause!");
        }
    }
    if ( selectQuery.getQueryPattern() != null ) {
        if (!this.selectQuery.toString().contains(v)) {
            throw new Exception("Recommendation variable not contained in the SelectPart!");
        }
    }
}
/*
 * Sets the aggregation object and tests whether the SKOSRec query contains a postfilter SELECT query and
 * whether the aggregation variable is contained in the SELECT query
 * @param a the summarization method (MAX | SUM | AVG)
 * @param i the IRI resource of the user profile for which aggregation-based retrieval should be performed
 * @param v the aggregation variable
 */
public void setAggregation (String a, String i, String v) throws Exception {
    if (!this.hasSelectQuery()) {
        throw new Exception("The SKOSRec query does not contain a SelectPart!");
    }
    if (!this.selectQuery.getQueryPattern().toString().contains(v)) {
        throw new Exception("Aggregation variable not contained in the SelectPart!");
    }
    this.aggregation = new Aggregation(a,i,v);
}
// Determines whether the SKOSRec query contains an aggregation object
public boolean hasAggregation () {
    if (this.aggregation != null) { return true; }
    else { return false; }
}
// Returns the aggregation object
public Aggregation getAggregation () {
    return this.aggregation;
}
/*

```

```

    * Retrieves recommendations based on preference scores, preference querying and concept expansion
    * @param pref the preference score
    * @param var the graph pattern to retrieve preferred LOD resources
    * @param sim the similarity threshold for concept expansion
    */
    public void addVarPart (double pref, VarPart var, SimCondition sim) throws Exception {
        List<String> items = getItems(var);
        for (String item : items) {
            addItemPart(pref, item, sim);
        }
    }
    /*
    * Retrieves recommendations based on preference scores and preference querying
    * @param pref the preference score
    * @param var the graph pattern to retrieve preferred LOD resources
    */
    public void addVarPart (double pref, VarPart var) throws Exception {
        List<String> items = getItems(var);
        for (String item : items) {
            addItemPart(pref, item);
        }
    }
    /*
    * Retrieves recommendations based on preference querying and concept expansion
    * @param var the graph pattern to retrieve preferred LOD resources
    * @param sim the similarity threshold for concept expansion
    */
    public void addVarPart (VarPart var, SimCondition sim) throws Exception {
        List<String> items = getItems(var);
        for (String item : items) {
            addItemPart(item, sim);
        }
    }
    /*
    * Retrieves recommendations based on preference querying
    * @param var the graph pattern to retrieve preferred LOD resources
    */
    public void addVarPart (VarPart var) throws Exception {
        List<String> items = getItems(var);
        for (String item : items) {
            addItemPart(item);
        }
    }
    // Execution of preference querying: retrieval of the preferred LOD resources based on a specified graph pattern
    /*
    * @param varRaw the graph pattern to retrieve preferred LOD resources
    * @return list the list of preferred LOD resources
    */
    public List<String> getItems (VarPart varRaw) throws Exception {
        List<String> list = new ArrayList<String>();
        String where = varRaw.getWhere();
        String varWithQ = varRaw.getVar();
        String var = varWithQ.replace("?", "");
        String query = "SELECT "+varWithQ+" WHERE "+where;
        List<Map<String, Object>> results = this.conf.getContainer().getCollection().getService().query(query);
        for (Iterator<Map<String, Object>> iter = results.iterator(); iter.hasNext(); ) {
            Map<String, Object> element = iter.next();
            String item = element.get(var).toString();
            list.add(item);
        }
        return list;
    }
    /*
    * Adds a preference (LOD resource and similarity threshold) to the user profile
    * @param iri the IRI of the preferred item
    * @param sim the similarity threshold for concept expansion
    */
    public void addItemPart (String iri, SimCondition sim) {
        ItemPart itemPart = new ItemPart(iri);
        itemPart.setSimCondition(sim.getRel(), sim.getSim());
        itemPartList.add(itemPart);
    }
    /*
    * Adds a preference (preference score, LOD resource and similarity threshold) to the user profile
    * @param pref the preference score
    * @param iri the IRI of the preferred item
    * @param sim the similarity threshold for concept expansion
    */
    public void addItemPart (double pref, String iri, SimCondition sim) {

```

```

ItemPart itemPart = new ItemPart(iri);
itemPart.setSimCondition(sim.getRel(),sim.getSim());
itemPart.setPref(pref);
itemPartList.add(itemPart);
}
/*
 * Adds a preference (LOD resource) to the user profile
 * @param iri the IRI of the preferred item
 */
public void addItemPart (String iri) {
ItemPart itemPart = new ItemPart(iri);
itemPartList.add(itemPart);
}
/*
 * Adds a preference (LOD resource and similarity threshold) to the user profile
 * @param iri the IRI of the preferred item
 * @param sim the similarity threshold for concept expansion
 */
public void addItemPart (double pref, String iri) {
ItemPart itemPart = new ItemPart(iri);
itemPart.setPref(pref);
itemPartList.add(itemPart);
}
/*
 * Returns the user profile
 * @return itemPartList the user profile with information on preferred items
 */
public List<ItemPart> getItemParts () {return itemPartList;}
}

```

A.4 SKOSRec Compiler

```

package rec;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import utils.MapSorter;
import com.hp.hpl.jena.query.Query;
import com.hp.hpl.jena.sparql.core.Var;
import federation.Configuration;
import java.io.FileNotFoundException;

public class RecCompiler {
private String simName = "";
private String simVar = "";
private RecQuery rQuery = null;
private Configuration conf = null;
/*
 * Compiles the SKOSRec query and check its semantic correctness
 * @param query the SKOSRec query
 * @param conf the configuration
 * @return results the solution table of the SKOSRec query
 */
public List<Map<String, Object>> compile (RecQuery query, Configuration conf) throws NoDescriptorException, FileNotFoundException {
this.simVar = query.getSimProjection().getVar();
this.simName = simVar.replace("?", "");
this.rQuery = query;
this.conf = conf;
HashMap<String, Double> recommendations = preprocess();
List<Map<String, Object>> results = postprocess(recommendations);
return results;
}
/*
 * Preprocessing of the SKOSRec query and execution of similarity calculation
 * @return recommendations similar items for the user profile from the LOD repository
 */
private HashMap<String, Double> preprocess () throws NoDescriptorException {
// Determines the number of recommendations to be returned to the user
int limit = this.rQuery.getSimProjection().getLimit();
// Identifies the preferred LOD resources that were either specified by the user or obtained from preference querying
List<ItemPart> articles = rQuery.getItemParts();
/*
 * Checks whether the SKOSRec query contains a prefilter condition
 * If it does the condition is handed over to the process of similarity calculation
 */
String wherePart;

```



```

if (this.rQuery.hasWhereClause()) { wherePart = this.rQuery.getWhereClause(); }
else { wherePart = "";}
RecRecommender recommender = new RecRecommender(articles, this.conf.toBeOptimized());
return recommender.recommend(conf.getContainer(), this.simVar, this.simName, wherePart, limit, 1.0);
}
/*
 * Postprocessing of recommendations (i.e., similar items)
 * @return output Solution table of the SKOSRec query
 */
private List<Map<String, Object>> postprocess (HashMap<String, Double> recommendations) {
List<Map<String, Object>> output;
// In case the SKOSRec query does not contain a postfilter condition the recommendations are returned right away
if (!rQuery.hasSelectQuery()) {
output = new ArrayList<>();
for (String recommendation : recommendations.keySet()) {
Map<String, Object> resultMap = new HashMap<>();
resultMap.put(simVar, recommendation);
output.add(resultMap);
}
return output;
}
// Otherwise the set of similar items is joined with the postfilter condition
else {
Query selectQuery = this.rQuery.getSelectQuery();
String vals = "";
Var var = Var.alloc(simName);
List<Var> varList = new ArrayList<Var>();
varList.add(var);
for (String rec : recommendations.keySet()) {
vals = vals+ " <"+rec+"> ";
}
String whereString = selectQuery.getQueryPattern().toString().replace(this.simVar, " VALUES "+this.simVar+" { "+vals+" } "+simVar+" ");
String varString = "";
boolean containsVar = false;
// Includes the recommendation variable in the solution table in case it is not specified
for (Var v : selectQuery.getProjectVars()) {
if (v.equals(this.simVar)) {
containsVar = true;
break;
}
else {
varString = varString + " "+v.toString();
}
}
}
if (containsVar == false) {
varString = varString + " "+this.simVar;
}
String selectString = "SELECT ";
if (selectQuery.isDistinct()) {
selectString = selectString+" DISTINCT ";
}
else if (selectQuery.isReduced()) {
selectString = selectString+" REDUCED ";
}
String queryString = selectString+varString+" WHERE "+whereString+' '+rQuery.getSelectQuery().getLimit();
// Retrieves the intermediate postfiltered solution table
List<Map<String, Object>> intermediate = this.conf.getContainer().getCollection().query(queryString);
// In case the SKOSRec query does not contain an aggregation section the intermediate solution table is returned right away
if (!rQuery.hasAggregation()) {
output = intermediate;
return output;
}
// Otherwise the solution table is sorted according to aggregation scores
else {
output = MapSorter.sortByList(intermediate, recommendations.keySet());
HashMap<String, HashMap<String, Double>> aggMap = new HashMap<>();
HashMap<String, Double> aggMapScore = new HashMap<String, Double>();
String aggVar = rQuery.getAggregation().getVar().replace("?", "");
String aggIRI = rQuery.getAggregation().getIRI();
String method = rQuery.getAggregation().getType();
for (Map<String, Object> mapAgg : output) {
String recomb = mapAgg.get(simName).toString();
String agg = mapAgg.get(aggVar).toString();
Double score = recommendations.get(recomb);
if (!aggMap.containsKey(agg)) {
HashMap<String, Double> recMap = new HashMap<>();
recMap.put(recomb, score);
aggMap.put(agg, recMap);
}
}
}
}

```

```

else {
    HashMap<String,Double>tempRec = aggMap.get(agg);
    tempRec.put(recomm, score);
    aggMap.put(agg, tempRec);
}
}

// Summarizes aggregation scores according to the specification in the SKOSRec query (MAX, SUM or AVG)
for (String agg : aggMap.keySet()) {
    switch (method) {
    case "MAX":
        Double max = 0.0;
        for (Double maxCand : aggMap.get(agg).values()) {
            if (maxCand > max) {
                max = maxCand;
            }
        }
        aggMapScore.put(agg, max);
        break;
    case "SUM":
        Double valSum = 0.0;
        for (Double val : aggMap.get(agg).values()) {
            valSum = valSum + val;
        }
        aggMapScore.put(agg, valSum);
        break;
    case "AVG":
        Double sum = 0.0;
        Double mean = 0.0;
        Double counter = 0.0;
        for (Double val : aggMap.get(agg).values()) {
            sum = sum + val;
            counter++;
        }
        mean = sum / counter;
        aggMapScore.put(agg, mean);
        break;
    }
}
}

Map<String,Double> sortedAggs = MapSorter.sortByValue(aggMapScore);
output = MapSorter.sortByList(intermediate, sortedAggs.keySet());
// Determines aggregation-based recommendations
List<String> aggregations = new ArrayList<>();
int counter = 0;
for (Map.Entry<String, Double> entry : sortedAggs.entrySet()) {
    if (!entry.getKey().equals(aggIRI)) {
        if (counter < rQuery.getSelectQuery().getLimit()){
            aggregations.add(entry.getKey());
            counter++;}
        else {break;}
    }
}
// Returns result mappings for aggregation-based recommendations
List<Map<String,Object>> newOutput = new ArrayList<>();
for (Map<String,Object> map : output) {
    if (aggregations.contains(map.get(aggVar).toString())) {
        newOutput.add(map);
    }
}
return newOutput;
}
}
}
}

```

A.5 Similarity Calculation

```

package rec;
import federation.CollectionContainer;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import federation.QueryRecommenderJob
/**
 * This class performs similarity calculation
 * @param articles LOD resources of the user profile
 * @param op indicates whether the retrieval process should be optimized
 * @param ranker object to order similar items

```

```

* @param mapSourceToTarget maps SKOS concepts of the source collection to SKOS concepts of the target collection
* @param mapTargetToSource maps SKOS concepts of the target collection to SKOS concepts of the source collection
*/
public class RecRecommender {
    List<ItemPart> articles;
    boolean op;
    RecRanker ranker;
    HashMap<String,String> mapSourceToTarget;
    HashMap<String,String> mapTargetToSource;
    HashMap<String,Double> scores = new HashMap<>();

    public RecRecommender (List<ItemPart> arts, boolean opt) {
        this.articles = arts;
        this.op = opt;
    }
    /*
    * @param coll recommendations are retrieved from this collection
    * @param simVar the recommendation variable
    * @param simName the recommendation variable as string value
    * @param whereCond the prefilter condition
    * @param limit the number of recommendations to be retrieved
    * @return a ordered list of LOD resources ranked according to the recommendation score
    */
    public HashMap<String,Double> recommend (CollectionContainer coll, String simVar,
        String simName, String whereCond, int limit) throws NoDescriptorException {
        // Initializes process that handles SPARQL API access operations
        QueryRecommenderJob job = new QueryRecommenderJob(coll, simVar, simName, whereCond);
        ArrayList<String> articleStrings = new ArrayList<>();
        /*
        * Retrieves similar items for each LOD resource in the user profile
        * In case the profile contains a similarity threshold the system retrieves additional proximate descriptors
        */
        for (ItemPart article : this.articles) {
            String resource = article.getIRI();
            articleStrings.add(resource);
            List<String> exactSubjects = job.getConcepts(resource);
            HashMap<String,Double> simSubjects = null;
            HashMap<String,Integer> allCounts;
            List<String> mergedList = new ArrayList<>(exactSubjects);
            if ( article.hasSim() ) {
                simSubjects = job.getProximateConceptsAndSim(exactSubjects, article.getSimCondition().getRel(), article.getSimCondition().getSim());
                List<String> proxies = job.getProximateConcepts(simSubjects);
                mergedList.addAll(proxies);
            }
            // In case the process is triggered in cross-repository retrieval mode, SKOS mappings need to be obtained
            if ( job.isDistributedJob() ) {
                HashMap<String,String> mapSourceToTarget = job.getMapSourceToTarget(mergedList);
                HashMap<String,String> mapTargetToSource = job.getMapTargetToSource();
                this.ranker = new RecRanker (mapSourceToTarget, mapTargetToSource);
                List<String> remoteConceptList = new ArrayList<>();
                for (String entry : mapTargetToSource.keySet()) {
                    remoteConceptList.add(entry);
                }
                allCounts = job.getConceptCounts(remoteConceptList);
            }
            else {
                this.ranker = new RecRanker();
                allCounts = job.getConceptCounts(mergedList);
            }
            // In case no SKOS concepts could be found for the LOD resource continue with the next resource in the profile
            if (allCounts.isEmpty()) {continue;}
            /*
            * Obtains information content (IC) scores for each SKOS annotation
            * In case the process is executed in optimized mode a cut value is determined
            */
            HashMap<String,Double> ICValues = job.getICScores(allCounts);
            String cutPart = "";
            int cut;
            if (this.op) {
                int noOfMatches = job.getCutBegin(limit);
                RecOptimizer helper;
                if (job.isDistributedJob()) {
                    helper = new RecOptimizer(mapSourceToTarget, mapTargetToSource);
                }
                else { helper = new RecOptimizer();}
                cut = helper.getCutOff(exactSubjects, ICValues, simSubjects, noOfMatches);
                cutPart = "having (COUNT(?subject) >= "+cut+" ) ";
            }
            HashMap<String,List<String>> simResults = job.getConceptsOfResources(cutPart);

```

```

HashMap<String,Double> localScores;
// Determines the recommendation score of each retrieved LOD resources
localScores = ranker.score(simResults, simSubjects, ICValues, exactSubjects);
for ( String key : localScores.keySet() ) {
  if (!this.scores.containsKey(key)){
    if(article.hasPref()) {
      this.scores.put(key, localScores.get(key)*article.getPref());
    }
    else { this.scores.put(key, localScores.get(key));}
  }
  else {
    double newScore;
    if(article.hasPref()) {
      newScore = this.scores.get(key) + (localScores.get(key)*article.getPref());
    }
    else { newScore = this.scores.get(key) + localScores.get(key); }
    this.scores.put(key, newScore); }
  }
}
// Ranks the LOD resources according to the recommendation score
return this.ranker.rank(this.scores, limit, articleStrings);
}
}

```

A.6 Ranker

```

package rec;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import utils.MapSorter;
/**
 * @param sourceMapping SKOS mappings from source to target collection (cross-repository retrieval)
 * @param targetMapping SKOS mappings from target to source collection (cross-repository retrieval)
 * @param rankDistributed indicates whether this is a distributed retrieval process
 */
public class RecRanker {
  HashMap<String,String> sourceMapping;
  HashMap<String,String> targetMapping;
  private boolean rankDistributed;
  // Constructor for a retrieval process that is executed over the default repository
  public RecRanker () {
    this.rankDistributed = false;
  }
  // Constructor for a cross-repository retrieval process
  public RecRanker (HashMap<String,String> sourceMapping, HashMap<String,String> targetMapping) {
    this.sourceMapping = sourceMapping;
    this.targetMapping = targetMapping;
    this.rankDistributed = true;
  }
  /*
   * Determines the recommendation score of retrieved LOD resources for input resources
   * @param results relevant resources and annotations
   * @param subjects proximate concepts and their concept-to-concept similarity score
   * @param ICVals IC scores of each SKOS concept
   * @param SKOS concepts of the input resource
   */
  public HashMap<String,Double> score (Map<String,List<String>> results, Map<String,Double> subjects,
  HashMap<String,Double> ICVals, List<String> exactSubjects) {
    HashMap<String,Double> sc = new HashMap<>();
    for (Map.Entry<String, List<String>> entry2 : results.entrySet() ) {
      List<String> matches;
      if (this.rankDistributed) {
        matches = new ArrayList<>();
        for ( String match : entry2.getValue() ) {
          matches.add(this.targetMapping.get(match));
        }
      }
      else { matches = entry2.getValue(); }
      String art = entry2.getKey();
      double score = 0.0;
      for (String sub : exactSubjects){
        double ICValue = 0.0;
        if (rankDistributed) {
          if ((this.sourceMapping.get(sub) != null) && (ICVals.get(this.sourceMapping.get(sub)) != null) ) {
            ICValue = ICVals.get(this.sourceMapping.get(sub));
          }
        }
      }
    }
  }
}

```

```

    }
  }
  else { if ( ICVals.get(sub) != null) { ICValue = ICVals.get(sub); } }
  if (matches.containsKey(sub)){ score = score + ICValue; }
  else if (subjects != null) {
    double proxSim = 0.0;
    for (String match : matches) {
      if (subjects.containsKey(sub+"-"+match)) {
        double newSim = subjects.get(sub+"-"+match);
        if (newSim >= proxSim){
          proxSim = newSim;
        }
      }
    }
    score = score + (proxSim * ICValue);
  }
}
sc.put(art, score);
}
return sc;
}
/*
 * Ranks the LOD resources according to the recommendation score
 * @param scores recommendation scores for LOD resources
 * @param limit the number of recommendations that should be retrieved
 * @param articles the LOD resources of the user profile
 * @return sortedRecs similar LOD resources (recommendations)
 */
public HashMap<String,Double> rank (HashMap<String,Double> scores, Integer limit, List<String> articles) {
  Map<String,Double> sortedMap = MapSorter.sortByValue(scores);
  int counter = 0;
  HashMap<String,Double> rec = new HashMap<>();
  for (Map.Entry<String, Double> entry : sortedMap.entrySet()) {
    if (!articles.contains(entry.getKey())) {
      if (counter < limit){
        rec.put(entry.getKey(),entry.getValue());
        counter++;
      }
      else { break; }
    }
  }
  HashMap<String,Double> sortedRecs = (HashMap) MapSorter.sortByValue(rec);
  return sortedRecs;
}
}

```

A.7 Optimizer

```

package rec;
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Map.Entry;
/**
 * Optimizes similarity calculation
 * @param conceptScores object to store potential recommendation scores
 * @param sourceMapping SKOS mappings from the source to the target collection
 * @param targetMapping SKOS mappings from the target to the source collection
 * @param optDistributed indicates whether the optimization procedure is carried out for cross-repository retrieval
 */
public class RecOptimizer {
  List<Double> conceptScores;
  HashMap<String,String> sourceMapping;
  HashMap<String,String> targetMapping;
  private boolean optDistributed;

  public RecOptimizer () {}

  public RecOptimizer (HashMap<String,String> sourceMapping, HashMap<String,String> targetMapping) {
    this.sourceMapping = sourceMapping;
    this.targetMapping = targetMapping;
    this.optDistributed = true;
  }
  /*
   * Computes the cut value that reduces the number of records to be processed

```

```

* @param exactSubjects SKOS annotations of the input resource from the user profile
* @param IC information content (IC) scores of SKOS concepts
* @param proxSim proximate concepts and concept-to-concept similarity scores
* @param matches number of matching SKOS concepts
* @return cutOff returns the maximum number of matching SKOS concepts required for similarity calculation
*/
public int getCutOff(List<String> exactSubjects, HashMap<String,Double> IC, Map<String,Double> proxSim, int matches) {
    conceptScores = new ArrayList<>();
    for (Entry<String,Double> entry : IC.entrySet()) {
        conceptScores.add(entry.getValue());
    }
    if ( proxSim != null) {
        for (Entry<String,Double> entry2 : proxSim.entrySet()) {
            String id = entry2.getKey();
            double simValue = entry2.getValue();
            String [] parts = id.split("-");
            String exact = parts[0];
            double ICValue;
            double score;
            if (optDistributed) {
                if ( this.sourceMapping.get(exact) != null && IC.get(this.sourceMapping.get(exact)) != null ) {
                    ICValue = IC.get(this.sourceMapping.get(exact));
                    score = ICValue * simValue;
                    conceptScores.add(score);
                }
            }
            else {
                if (IC.get(exact) != null) {
                    ICValue = IC.get(exact);
                    score = ICValue * simValue;
                    conceptScores.add(score);
                }
            }
        }
    }
    Collections.sort(conceptScores);
    double prevMinScore;
    double maxScore;
    int cutOff = 0;
    while (matches > 0) {
        prevMinScore = getScore(matches, false);
        cutOff = matches;
        matches--;
        maxScore = getScore(matches, true);
        if ( maxScore < prevMinScore ) {
            break;
        }
    }
    return cutOff;
}
/*
* Determines potential recommendation scores for a given number of matching SKOS concepts
* @param limit number of matching SKOS concepts
* @param maxScores indicates if the function should calculate the potential for a higher (maxScore == TRUE)
* or a lower number of matching SKOS concepts (maxScore == FALSE)
* @return sum the potential recommendation score
*/
private double getScore(int limit, boolean maxScore) {
    double sum = 0;
    List<Double> subList;
    if (maxScore == true) {
        subList = conceptScores.subList((conceptScores.size()-1)-limit, conceptScores.size()-1);
    }
    else { subList = conceptScores.subList(0, limit); }
    for (Double value : subList) {
        sum = sum + value;
    }
    return sum;
}
}

```

B Supplementary Material of the Evaluation

B.1 Simulation - Flexible Similarity Detection

```

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import org.apache.mahout.cf.taste.common.NoSuchItemException;
import org.apache.mahout.cf.taste.common.NoSuchUserException;
import org.apache.mahout.cf.taste.common.TasteException;
import org.apache.mahout.cf.taste.impl.common.LongPrimitiveIterator;
import org.apache.mahout.cf.taste.model.DataModel;
import org.apache.mahout.cf.taste.model.PreferenceArray;
import federation.DistributionalCalculator;
/**
 * Simulation setting for flexible similarity detection
 * {@value #LOG2} logarithm base 2
 * {@value #namespace} namespace of DBPedia resources
 * {@value #keyNS} namespace of DBPedia resources
 * {@value #filePath} path to store evaluation results
 * {@value #sep} line separator to store evaluation results
 * {@value #rounds} cross-validation rounds
 */
public class LDRecommender {
    private static final double LOG2 = Math.log(2.0);
    private static final String namespace = "http://dbpedia.org/resource/";
    private static final String keyNS = "http://dbpedia.org/resource/Category:";
    private static final String filePath = "src/log.txt";
    private static final String sep = System.getProperty("line.separator");
    private static final int top = 20;
    private static final int rounds = 5;

    public static void main(String[] args) throws IOException, TasteException, NoSuchItemException, NoSuchUserException, \newline
    InterruptedException, ClassNotFoundException {
        File file = new File(filePath);
        FileOutputStream fw = new FileOutputStream(file, true);
        PrintWriter out = new PrintWriter(fw);
        out.write("userid;precision;recall;novelty;diversity"+sep);
        for ( int i = 0; i < rounds; i++) {
            DataReader dr = new DataReader();
            // Reads test and training data
            DataModel test = dr.readModel("src/test"+i+".dat");
            DataModel train = dr.readModel("src/train"+i+".dat");
            String path = "src/MappingMovieLens.tsv";
            String subjectPath = "src/movieSubjects.ser";
            String countPath = "src/movieCounts.ser";
            Mapper map = new Mapper(path);
            // Reads SKOS annotations and concept counts from DBpedia
            FileInputStream fis = new FileInputStream(subjectPath);
            ObjectInputStream ois = new ObjectInputStream(fis);
            Map<String, List<String>> subjects = (Map<String, List<String>>)
            ois.readObject();
            ois.close();
            FileInputStream fis2 = new FileInputStream(countPath);
            ObjectInputStream ois2 = new ObjectInputStream(fis2);
            HashMap<String, Integer> counts = (HashMap<String, Integer>)
            ois2.readObject();
            ois2.close();
            // Determines information content (IC) scores of SKOS concepts

```

```

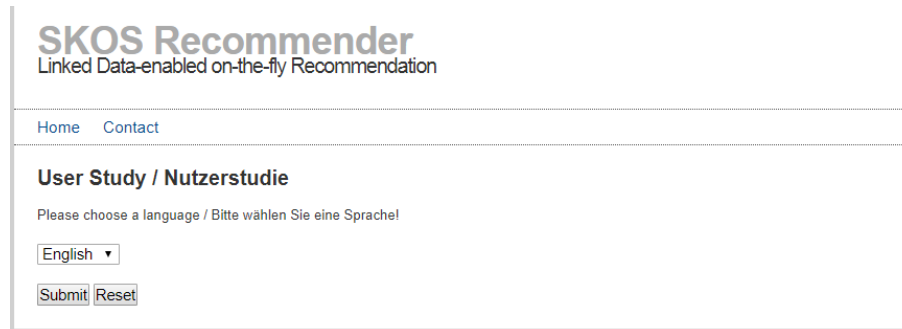
DistributionalCalculator dc = new DistributionalCalculator(1.0);
HashMap<String,Double> ICValues = dc.computeIC(counts);
// Initializes an object that can calculate diversity scores for recommendation lists
Diversity diversity = new Diversity(subjects,ICValues);
// Initializes an object that computes on-the-fly recommendations
FastRecommendation fastRec = new FastRecommendation(subjects, counts, namespace, keyNS, ICValues);
int totalNoOfPreferences = 0;
// Determines the total number of preferences stated by users
for (LongPrimitiveIterator itemIt = train.getItemIDs(); itemIt.hasNext();) {
    Long item = itemIt.nextLong();
    totalNoOfPreferences = totalNoOfPreferences + train.getNumUsersWithPreferenceFor(item);
}

PreferenceArray realPrefs = null;
int step = 0;
/*
 * Performs recommendation retrieval for the first 100 users in the dataset based on the preferences
 * in the training set and compares the recommendations with the actual preferences from the test set
 */
for (LongPrimitiveIterator userIt = test.getUserIDs(); userIt.hasNext();) {
    if (step < 100) {
        step++;
        double novItemSumREL = 0.0;
        double novelty = 0.0;
        Long userID = userIt.nextLong();
        realPrefs = test.getPreferencesFromUser(userID);
        List<Long> recommendations = null;
        Map<String,Double> ids = new HashMap<String,Double>();
        for (LongPrimitiveIterator itemIt = train.getItemIDsFromUser(userID).iterator(); itemIt.hasNext();) {
            Long itemID = itemIt.nextLong();
            ids.put(map.getString(itemID),Double.valueOf(train.getPreferenceValue(userID, itemID)));
        }
        recommendations = fastRec.getRecommendations(ids, 1.0, top, map);
        List<String> stringRec = new ArrayList<String>();
        for (Long recomm : recommendations) {
            stringRec.add(map.getString(recomm));
        }
        int matches = 0;
        int precisionBase;
        float precision = 0.0f;
        float recall = 0.0f;
        if ( recommendations.size() < top) {
            precisionBase = recommendations.size();
        }
        else { precisionBase = top; }
        // Determines the number of matching resources of the set of predicted recommendations and the actual preferences
        for (int m = 0; m < precisionBase; m++) {
            if (realPrefs.hasPrefWithItemID(recommendations.get(m))) {
                matches++;
                double fraction2 = (double) train.getNumUsersWithPreferenceFor(recommendations.get(m)) / (double) totalNoOfPreferences;
                if (fraction2 == 0.0) {
                    fraction2 = 1 / (double) totalNoOfPreferences;
                }
                novItemSumREL = novItemSumREL + (-Math.log(fraction2)/LOG2);
            }
        }
        // Calculation of precision, recall, novelty and diversity scores
        if (precisionBase > 0) {
            precision = (float) matches / (float) precisionBase;
            recall = (float) matches / (float) realPrefs.length();
            novelty = novItemSumREL / (float) matches;
        }
        double div = diversity.calcDiv(stringRec);
        // Stores the evaluation results in a log file
        out.write(userID+";"+precision+";"+recall+";"+novelty+";"+div+sep);
        out.flush();
    }
}
}
out.close();
}
}

```

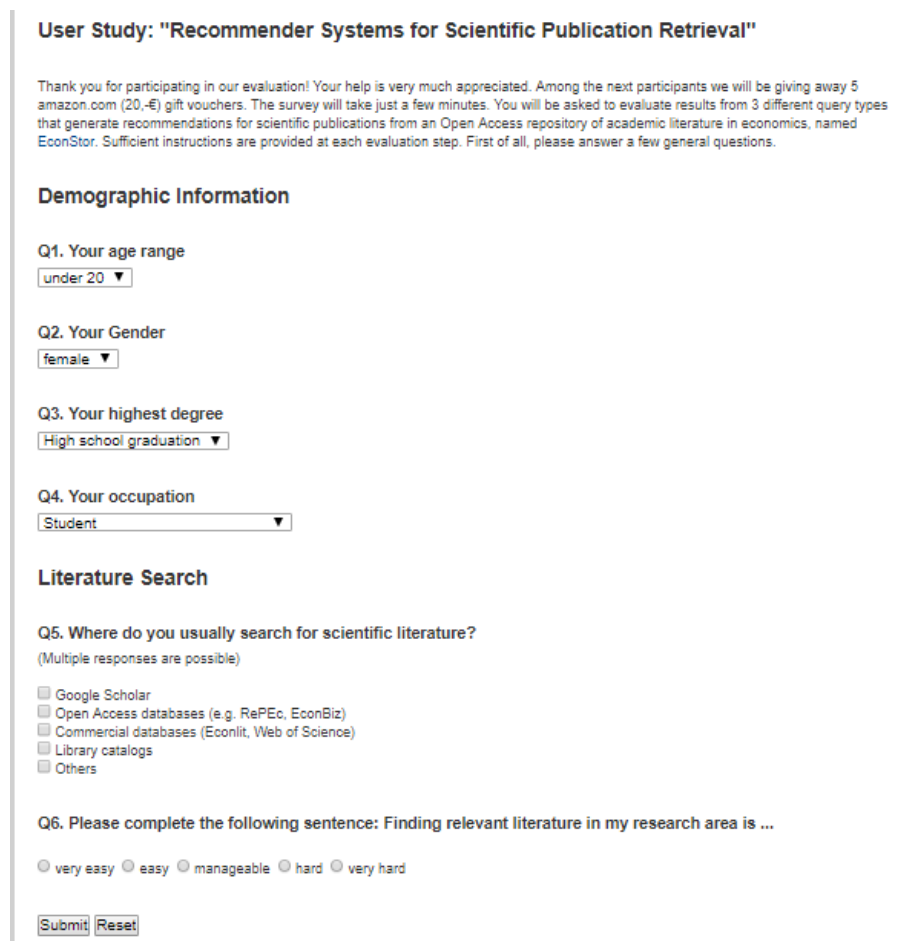

B.2 Webinterfaces of the Online Studies

B.2.1 Digital Library Experiment



The screenshot shows the top part of a web interface. At the top left, it says "SKOS Recommender" in a large font, with "Linked Data-enabled on-the-fly Recommendation" underneath. Below this is a navigation bar with "Home" and "Contact" links. The main heading is "User Study / Nutzerstudie". Below that, there is a prompt: "Please choose a language / Bitte wählen Sie eine Sprache!". There is a dropdown menu currently showing "English". At the bottom of the form are two buttons: "Submit" and "Reset".

Fig. B.1: DL - Language selection (page 1)



The screenshot shows a survey page titled "User Study: 'Recommender Systems for Scientific Publication Retrieval'". It starts with a thank-you message and an incentive: "Thank you for participating in our evaluation! Your help is very much appreciated. Among the next participants we will be giving away 5 amazon.com (20,-€) gift vouchers. The survey will take just a few minutes. You will be asked to evaluate results from 3 different query types that generate recommendations for scientific publications from an Open Access repository of academic literature in economics, named EconStor. Sufficient instructions are provided at each evaluation step. First of all, please answer a few general questions."

The "Demographic Information" section contains four questions, each with a dropdown menu:

- Q1. Your age range: under 20
- Q2. Your Gender: female
- Q3. Your highest degree: High school graduation
- Q4. Your occupation: Student

The "Literature Search" section contains two questions:

- Q5. Where do you usually search for scientific literature? (Multiple responses are possible). This is a checkbox question with options: Google Scholar, Open Access databases (e.g. RePEc, EconBiz), Commercial databases (Econlit, Web of Science), Library catalogs, and Others.
- Q6. Please complete the following sentence: Finding relevant literature in my research area is ... This is a radio button question with options: very easy, easy, manageable, hard, very hard.

At the bottom of the form are "Submit" and "Reset" buttons.

Fig. B.2: DL - Demographics section (page 2)

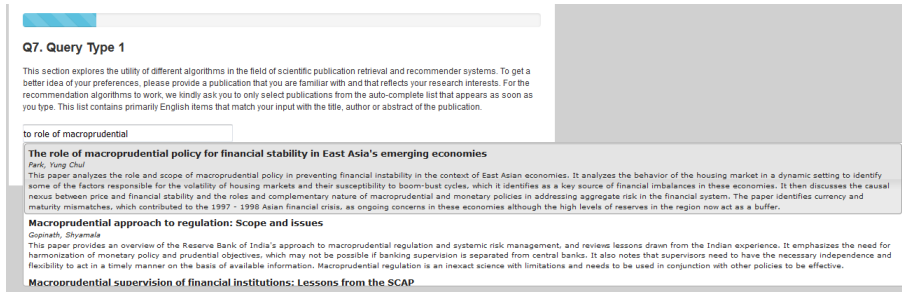


Fig. B.3: DL - User profile generation, TC1 (page 3)

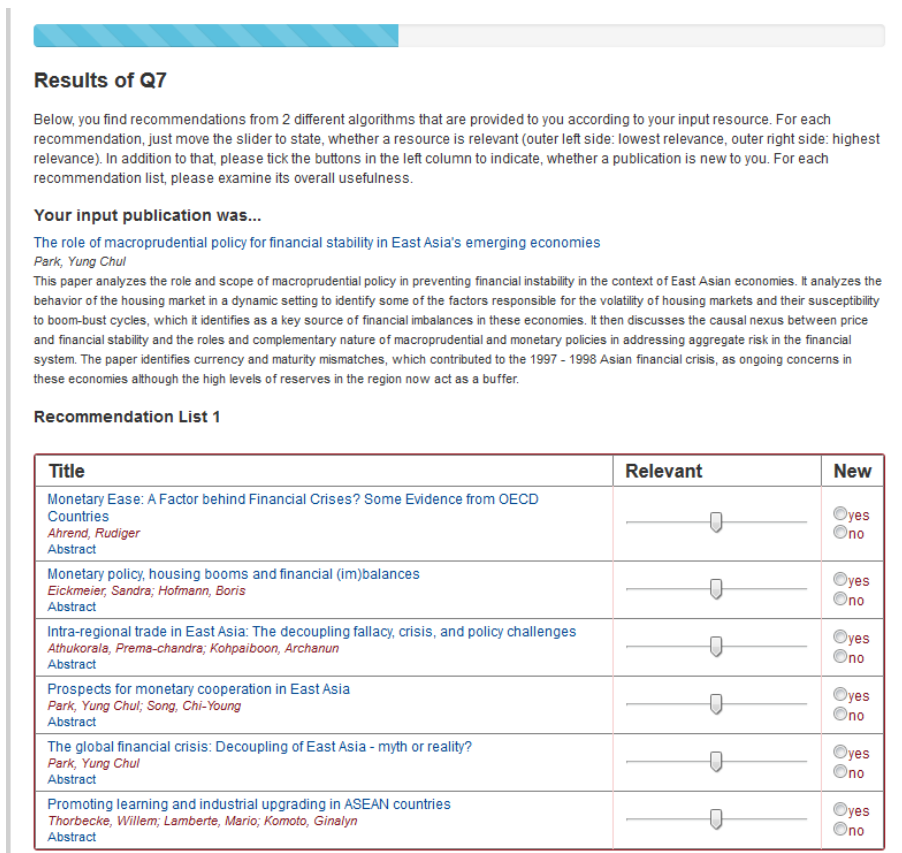


Fig. B.4: DL- Results part I, TC1 (page 4)

Statement	Agreement
The recommendations of List 1 better fit my research interests than what I may receive from a research fellow.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The publications of List 1 are diverse.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
I feel supported to find relevant publications with the help of recommendations of List 1	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree

Recommendation List 2

Title	Relevant	New
Toward a regional exchange rate regime in East Asia <i>Kawai, Masahiro</i> Abstract	<input type="range"/>	<input type="radio"/> yes <input type="radio"/> no
Towards a macroprudential surveillance and remedial policy formulation system for monitoring financial crisis <i>Bhattacharyay, Biswa N.</i> Abstract	<input type="range"/>	<input type="radio"/> yes <input type="radio"/> no
Crisis, imbalances, and India <i>Kumar, Rajiv; Vashisht, Pankaj</i> Abstract	<input type="range"/>	<input type="radio"/> yes <input type="radio"/> no
The role of macroeconomic policy in rebalancing growth <i>Morgan, Peter J.</i> Abstract	<input type="range"/>	<input type="radio"/> yes <input type="radio"/> no

Statement	Agreement
The recommendations of List 2 better fit my research interests than what I may receive from a research fellow.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The publications of List 2 are diverse.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
I feel supported to find relevant publications with the help of recommendations of List 2	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree

Fig. B.5: DL - Results part II, TC1 (page 4)

Q8. Query Type 2

This section examines further recommendation algorithms. Please select a publication that you are familiar with and that reflects your research interests. You can either choose the publication you used in the previous section (Q7) or a new one. For the recommendation algorithms to work, we kindly ask you to only enter publications from the auto-complete list that appears as soon as you type. This list contains primarily English and German items that match your input with the title, author or abstract of the publication.

Donor support

Donors' support for microcredit as social enterprise: A critical reappraisal
Wasank, Ruchko
The donor community has enthusiastically embraced the concept of microfinance as a promising mechanism to attain the objectives of poverty alleviation and microenterprise development. Amid the high expectation, a myth has been inadvertently created that they could be the ultimate solution to poverty reduction. The objective of the paper is to examine the nature of support rendered by the donor community to microfinance programmes and the effectiveness of this particular outlet of official aid for microenterprise development and poverty alleviation. To this end, the paper first examines economics of microfinance as an instrument of microenterprise development and poverty reduction as well as its delivery mechanisms. The paper then assesses empirical evidence of the performance of microfinance institutions and their impacts on poverty alleviation and microenterprise development. Given this background, the paper discusses main features and trends in donor support and policy implication of the analysis for the donor community.

Donor coordination and specialization: Did the Paris declaration make a difference?
Numenkamp, Peter; Ohler, Hannes; Thiele, Rainer
We assess whether bilateral and multilateral donors of foreign aid specialized and coordinated their activities with other donors as agreed in the Paris Declaration of 2005. We account for donor heterogeneity, varying aid priorities and recipient characteristics in order to isolate changes in donor behaviour over time. Recent shifts in aid priorities, such as the rising importance of general budget support, have reduced the fragmentation of aid. Nevertheless, our results reveal that aid fragmentation persisted after the Paris Declaration and coordination among donors has even weakened.

Fig. B.6: DL - User profile generation, TFFlex (page 5)

Results of Q8

Below, you find recommendations from 3 different algorithms that are provided to you according to your input resource. For each recommendation, just move the slider to state, whether a resource is relevant (outer left side: lowest relevance, outer right side: highest relevance). Please tick the buttons in the left column to indicate, whether a publication is new to you. In addition to that, please rank the recommendation lists according to their usefulness and diversity (see bottom of the page).

Your input publication was...

Donors' support for microcredit as social enterprise: A critical reappraisal
Nissanke, Machiko
 The donor community has enthusiastically embraced the concept of microfinance as a promising mechanism to attain the objectives of poverty alleviation and microenterprise development. Amid the high expectation, a myth has been inadvertently created that they could be the ultimate solution to poverty reduction. The objective of the paper is to examine the nature of support rendered by the donor community to microfinance programmes and the effectiveness of this particular outlet of official aid for microenterprise development and poverty alleviation. To this end, the paper first examines economics of microfinance as an instrument of microenterprise development and poverty reduction as well as its delivery mechanisms. The paper then assesses empirical evidence of the performance of microfinance institutions and their impacts on poverty alleviation and microenterprise development. Given this background, the paper discusses main features and trends in donor support and policy implication of the analysis for the donor community.

Recommendation List 1

Title	Relevant	New
Ownership and the donor-recipient relationship <i>Whitfield, Lindsay</i> Abstract		<input type="radio"/> yes <input type="radio"/> no
Targeting aid to the needy and deserving : nothing but promises? <i>Nunnenkamp, Peter, Thiele, Rainer</i> Abstract		<input type="radio"/> yes <input type="radio"/> no
The Multilateral Donor Non-System: Towards Accountability and Efficient Role Assignment <i>Reisen, Helmut</i> Abstract		<input type="radio"/> yes <input type="radio"/> no
Development strategies and foreign aid policies for low income countries in the 1990s <i>Hiemenz, Ulrich</i> Abstract		<input type="radio"/> yes <input type="radio"/> no
Improving donor support for urban poverty reduction <i>Banks, Nicola</i> Abstract		<input type="radio"/> yes <input type="radio"/> no
Donor coordination and specialization: Did the Paris declaration make a difference? <i>Nunnenkamp, Peter, Ohler, Hannes; Thiele, Rainer</i> Abstract		<input type="radio"/> yes <input type="radio"/> no

Recommendation List 2

Title	Relevant	New
Lessons to be learned: Political party research and political party assistance <i>Erdmann, Gero</i> Abstract		<input type="radio"/> yes <input type="radio"/> no
Technological readiness in the Middle East and North Africa: Implications for Egypt <i>Brach, Juliane</i> Abstract		<input type="radio"/> yes <input type="radio"/> no
Aid and growth: What meta-analysis reveals <i>Mekasha, Tseday Jemaneh; Tarp, Finn</i> Abstract		<input type="radio"/> yes <input type="radio"/> no

Fig. B.7: DL - Results part I, TFlex (page 6)

Foreign assistance in a climate-constrained world <i>Amdt, Channing; Bach, Christian Friis</i> Abstract		<input type="radio"/> yes <input type="radio"/> no
Using indicators to encourage development: Lessons from the millennium development goals <i>Manning, Richard</i> Abstract		<input type="radio"/> yes <input type="radio"/> no
Lopsided business partnerships: An ex-post study of Danida's Private Sector Development Programme in India <i>Folke, Steen</i> Abstract		<input type="radio"/> yes <input type="radio"/> no

Recommendation List 3

Title	Relevant	New
Grants versus loans : much ado about (almost) nothing <i>Nunnenkamp, Peter; Thiele, Rainer; Wilfer, Tom</i> Abstract		<input type="radio"/> yes <input type="radio"/> no
Assessing the allocation of aid : developmental concerns and the self-interest of donors <i>Canavire-Bacameza, Gustavo; Nunnenkamp, Peter; Thiele, Rainer; Triveño, Luis</i> Abstract		<input type="radio"/> yes <input type="radio"/> no
Mehr ist nicht genug : wirksame Entwicklungshilfe für Afrika? <i>Nunnenkamp, Peter</i> Abstract		<input type="radio"/> yes <input type="radio"/> no
Elite capture, political voice and exclusion from aid: an experimental study <i>D'Exelle, Ben; Riedl, Arno</i> Abstract		<input type="radio"/> yes <input type="radio"/> no
The costs of favoritism: Is politically-driven aid less effective? <i>Dreher, Axel; Klasen, Stephan; Vreeland, James Raymond; Werker, Eric</i> Abstract		<input type="radio"/> yes <input type="radio"/> no
The Contribution of the IMF and the World Bank to Economic Freedom <i>Bookmann, Bernhard; Dreher, Axel</i> Abstract		<input type="radio"/> yes <input type="radio"/> no

Please rank the recommendation lists in terms of diversity. (Rank 1 represents the highest degree of diversity and Rank 3 the lowest.)

List 1

List 2

List 3

Please rank the recommendation lists in terms of their general usefulness. (Rank 1 represents the highest degree of usefulness and Rank 3 the lowest.)

List 1

List 2

List 3

Fig. B.8: DL - Results part II, TFlex (page 6)

[Home](#) [Contact](#)

Q9. Query Type 3

Thank you for your help so far! The last query type allows you to obtain constraint-based recommendations. Select the publication that you have already chosen in the last section (Q8). Please use at least one other tab ("Language" and/or "Constraint"), to narrow your search according to your information needs. For the recommendation algorithm to work, we kindly ask you to only select items that appear in the auto-complete lists of the two fields "Publication" and "Constraint". In the field "Constraint" you can choose a series (e. g. Kiel Economic Policy Papers) a keyword of the [STW Thesaurus for Economics](#) or an author.

Publication

Language

German English All languages

Constraint

STW Keyword

- Poverty
- Rural poverty
- Poverty reduction
- Urban poverty
- Child poverty
- Elderly poverty
- Poverty alleviation
- Old-age poverty

Fig. B.9: DL - User profile generation, TC2 (page 7)

Results of Q9

Below, you find recommendations that fit your input query. For each recommendation, just move the slider to state, whether a resource is relevant (outer left side: lowest relevance, outer right side: highest relevance). In addition to that, please tick the buttons in the left column to indicate, whether a publication is new to you. For each recommendation list, please examine its overall usefulness. When you are done, please press "Submit and finish" to complete the survey.

Your input publication was...

[Donors' support for microcredit as social enterprise: A critical reappraisal](#)
Nissanke, Machiko

The donor community has enthusiastically embraced the concept of microfinance as a promising mechanism to attain the objectives of poverty alleviation and microenterprise development. Amid the high expectation, a myth has been inadvertently created that they could be the ultimate solution to poverty reduction. The objective of the paper is to examine the nature of support rendered by the donor community to microfinance programmes and the effectiveness of this particular outlet of official aid for microenterprise development and poverty alleviation. To this end, the paper first examines economics of microfinance as an instrument of microenterprise development and poverty reduction as well as its delivery mechanisms. The paper then assesses empirical evidence of the performance of microfinance institutions and their impacts on poverty alleviation and microenterprise development. Given this background, the paper discusses main features and trends in donor support and policy implication of the analysis for the donor community.

Recommendation List

Title	Relevant	New
Shooting the messenger of good news : a critical look at the World Bank's success story of effective aid <i>Nunnenkamp, Peter</i> Abstract	<input type="range"/>	<input type="radio"/> yes <input type="radio"/> no
Targeting aid to the needy and deserving : nothing but promises? <i>Nunnenkamp, Peter; Thiele, Rainer</i> Abstract	<input type="range"/>	<input type="radio"/> yes <input type="radio"/> no
Halving poverty by doubling aid : how well founded is the optimism of the World Bank? <i>Langhammer, Rolf J.</i> Abstract	<input type="range"/>	<input type="radio"/> yes <input type="radio"/> no
Are NGOs the better donors? A case study of aid allocation for Sweden <i>Dreher, Axel; Mölders, Florian; Nunnenkamp, Peter</i> Abstract	<input type="range"/>	<input type="radio"/> yes <input type="radio"/> no
Sectoral aid priorities: Are donors really doing their best to achieve the millennium development goals? <i>Thiele, Rainer; Nunnenkamp, Peter; Dreher, Axel</i> Abstract	<input type="range"/>	<input type="radio"/> yes <input type="radio"/> no
Improving donor support for urban poverty reduction <i>Banks, Nicola</i> Abstract	<input type="range"/>	<input type="radio"/> yes <input type="radio"/> no

Statement	Agreement
The recommendations better fit my research interests than what I may receive from a research fellow.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The recommended publications are diverse.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
I feel supported to find relevant publications with the help of the recommendations.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The possibility to narrow the search with additional constraints helps me to get better recommendations.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree

Fig. B.10: DL - Results, TC2 (page 8)

SKOS Recommender

Linked Data-enabled on-the-fly Recommendation

[Home](#) [Contact](#)

Thank you!

You have finished the survey. Thank you very much for your participation. In case you want to win one of the amazon.com gift vouchers, please enter your e-mail address!

Feel free to [contact us](#) in case you have any questions!

Fig. B.11: DL - Finish screen (page 9)

Recommender Systems

User Studies

[Home](#) [Contact](#)

Travel Destination Search and Recommender Systems

Thank you for participating in our web-based experiment "Travel Destination Search and Recommender Systems"! Your help is very much appreciated. The study examines the usefulness of recommendation algorithms when searching for travel destinations. This includes the search for cities, regions, countries or special points of interest (e.g. monuments). Before taking part in this experiment, please read the following consent form and click on the "I Agree" button at the bottom of the page if you understand the statements and freely consent to participate in the study.

Consent Form

This web-based experiment is designed to understand preferences for different recommender systems' algorithms in the travel domain. The study is being conducted by Lisa Wenige (M.Sc.), research assistant at the Chair of Business Information Systems, Friedrich Schiller University Jena, Germany. No deception is involved, and the study involves no more than minimal risk to participants (i.e. the level of risk encountered in daily life).

Participation in the study typically takes 20 minutes and is strictly anonymous. Participants start by providing demographic information and answering a few questions regarding their travel search habits. Then, they will be asked to evaluate recommendation results in 3 different travel search test cases. Sufficient instructions are provided for each test case.

All responses are treated as confidential, and in no case will responses from individual participants be identified. Rather, all data will be published in aggregate form only. Participants should be aware, however, that the experiment is not being run from a "secure" https server of the kind typically used to handle credit card transactions, so there is a small possibility that responses could be viewed by unauthorized third parties (e.g. computer hackers).

Many individuals from a previously conducted similar experiment found it to be at least interesting, if not enjoyable, and no adverse reactions have been reported thus far. Participants from clickworker.com will receive a monetary compensation. Other visitors to this web site are welcome to complete the study, although they will receive no compensation. Participation is entirely voluntary. Participants are free to refuse to answer any question or withdraw from the experiment at any time. The decision about whether to take part in the study will not affect any future interactions of the participants with travel service providers or their relations with the Friedrich Schiller University Jena.

Participants who have further questions about this study or their rights, or wish to issue a complaint or concern may [contact us](#) via phone or e-mail.

If you are 18 years of age or older, understand the statements above, and freely consent to participate in the study, click on the "I Agree" button to begin the experiment.

Fig. B.12: Travel - Consent form (page 1)

B.2.2 Travel Experiment

[Home](#) [Contact](#)

Travel Destination Search and Recommender Systems

Please start by answering a few general questions about you and your travel search habits.

Demographic Information

Q1. Your age range

Q2. Your Gender

Travel

Q3. What type of vacations do you prefer?
(Multiple responses are possible)

- Adventure
- Beaches and Sun
- Mountains and Snow
- Countryside
- History and Culture
- City Trips
- Wellness
- Others

Q4. How often do you travel a year?

Q5. How long do you usually travel?

Travel Destination Search

Q6. How do you usually find out about interesting places and travel destinations?
(Multiple responses are possible)

- Family and Friends
- Local Travel Agency
- General Search Engines (e.g. Google, Bing)
- Online Travel Communities (e.g. Lonely Planet Forum, TravellersPoint)
- Online Travel Agencies (e.g. Booking.com, TripAdvisor)
- Others

Q7. Please complete the following sentence: Finding interesting travel destinations is ...

very easy easy manageable hard very hard

Please note: After pressing the submit button, it will take some time to load the following page.

Fig. B.13: Travel - Demographics section (page 2)

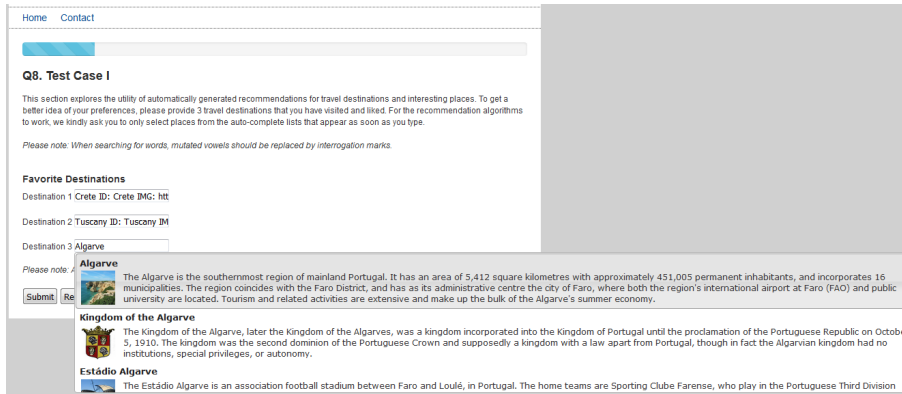


Fig. B.14: Travel - User profile generation, TC1 (page 3)

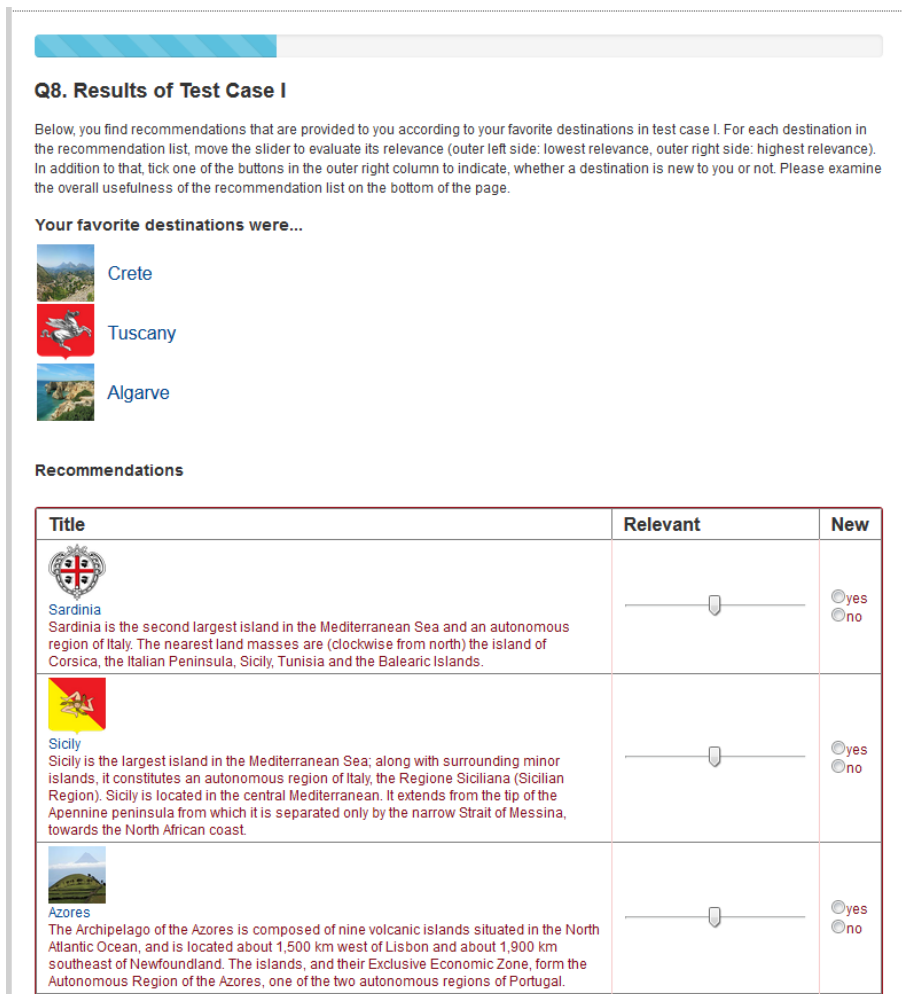







Fig. B.15: Travel - Results part I, TC1 (page 4)

 <p>Kea (island) Kea, also known as Gia or Tzia, Zea, and, in antiquity, Keos, is a Greek island in the Cyclades archipelago in the Aegean Sea. Kea is part of the Kea-Kythnos regional unit. Its capital, Ioulis, is inland at a high altitude (like most ancient Cycladic settlements, for fear of pirates) and is considered quite picturesque. Other major villages of Kea are the port of Korissia and the fishing village of Vourkari.</p>	<input type="checkbox"/> yes <input type="checkbox"/> no
 <p>Réunion Réunion is a French island with a population of about 800,000 located in the Indian Ocean, east of Madagascar, about 200 kilometres south west of Mauritius, the nearest island. Administratively, Réunion is one of the overseas departments of France. Like the other overseas departments, Réunion is also one of the 27 regions of France and an integral part of the Republic with the same status as those situated on the European mainland.</p>	<input type="checkbox"/> yes <input type="checkbox"/> no
 <p>Elba Elba is a Mediterranean island in Tuscany, Italy, 20 kilometres from the coastal town of Piombino. The largest island of the Tuscan Archipelago, Elba is also part of the Arcipelago Toscano National Park, and the third largest island in Italy, after Sicily and Sardinia. It is located in the Tyrrhenian Sea, about 50 kilometres east of the French island of Corsica.</p>	<input type="checkbox"/> yes <input type="checkbox"/> no
 <p>Ionian Islands The Ionian Islands are a group of islands in Greece. They are traditionally called the Heptanese, i.e. "the Seven Islands" (Greek: Επτάνησα, Heptanēsa or Επτάνησος, Heptanēsos; Italian: Eptaneso), but the group includes many smaller islands as well as the seven principal ones.</p>	<input type="checkbox"/> yes <input type="checkbox"/> no
 <p>Loulé Municipality Loulé is a city and a municipality in Portugal with a total area of 764.2 km² and a total population of 70,622 inhabitants. The city proper has a population of 24,791. The municipality is composed of 11 parishes, and is located in the District of Faro. The present Mayor is Sebastião Seruca Emídio, elected by the Social Democratic Party. The municipal holiday is Ascension Day.</p>	<input type="checkbox"/> yes <input type="checkbox"/> no

Statement	Agreement
The recommendations better fit my preferences than the suggestions I would receive from sources that I usually use to find out about interesting places (see Q6).	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The recommendations are diverse.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
I feel supported to find interesting travel destinations with the help of the recommendations.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree

Fig. B.16: Travel - Results part II, TC1 (page 4)

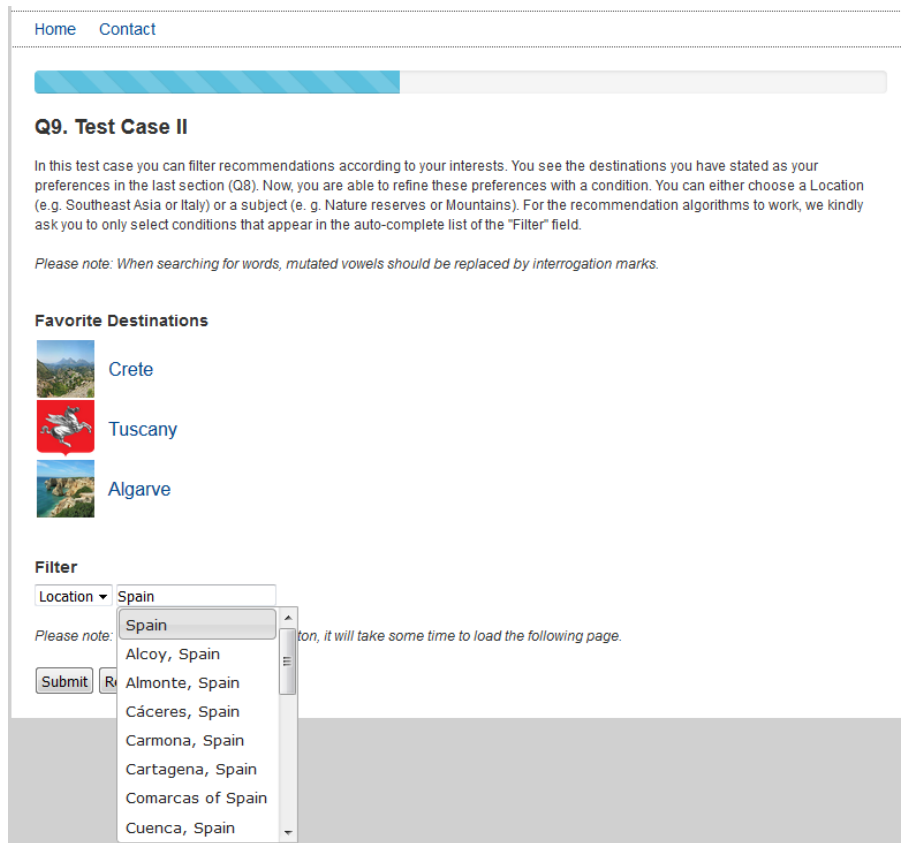





Fig. B.17: Travel - User profile generation, TC2 (page 5)

Q9. Results of Test Case II

Below, you find recommendations by different algorithms that fit your inputs in test case II. For each destination in one of the 2 recommendation lists, move the slider to evaluate its relevance (outer left side: lowest relevance, outer right side: highest relevance). In addition to that, tick one of the buttons in the outer right column to indicate, whether a destination is new to you or not. Please examine the overall usefulness of each recommendation list on the bottom of the list.

Your favorite places were...

-  Crete
-  Tuscany
-  Algarve

Your condition was: Spain

Recommendation List 1







Title	Relevant	New
 <p>Balearic Islands The Balearic Islands are an archipelago of Spain in the western Mediterranean Sea, near the eastern coast of the Iberian Peninsula. The four largest islands are: Majorca, Minorca, Ibiza and Formentera. The archipelago forms an autonomous community and a province of Spain, with Palma as the capital city. The co-official languages in the Balearic Islands are Catalan and Spanish. The current Statute of Autonomy declares the Balearic Islands as one nationality of Spain.</p>		<input type="radio"/> yes <input type="radio"/> no
 <p>Canary Islands The Canary Islands, also known as the Canaries, are a Spanish archipelago located just off the northwest coast of mainland Africa, 100 km west of the border between Morocco and the Western Sahara. The Canaries are one of Spain's 17 autonomous communities and an outermost region of the European Union.</p>		<input type="radio"/> yes <input type="radio"/> no
 <p>Peñón de Alhucemas Peñón de Alhucemas ("Lavender Rock") is one of the Spanish plazas de soberanía just off the Moroccan coast in the Alboran Sea. It is also one of several peñones, or rock-fortresses, on the coast of Northern Africa. Peñón de Alhucemas, together with the islets of "Isla de Mar" and "Isla de Tierra" slightly to the west, form the Alhucemas Islands.</p>		<input type="radio"/> yes <input type="radio"/> no

Fig. B.18: Travel - Results part I, TC2 (page 6)

Statement	Agreement
The recommendations of list 1 better fit my preferences than the suggestions I would receive from sources that I usually use to find out about interesting places (see Q6).	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The recommendations of list 1 are diverse.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
I feel supported to find interesting travel destinations with the help of the recommendations of list 1.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The filter has improved the recommendations of list 1.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree

Recommendation List 2





Title	Relevant	New
 <p>Albelda de Iregua Albelda de Iregua is a village in the province and autonomous community of La Rioja, Spain. The municipality covers an area of 23.03 square kilometres and as of 2011 had a population of 3339 people.</p>	<input type="range"/>	<input type="radio"/> yes <input type="radio"/> no
 <p>Grávalos Grávalos is a village in the province and autonomous community of La Rioja, Spain. It's 72 km. from the city of Logroño. Traditional tales talk about Grávalos like belong to Arnedo. It was in first times a Roman villa. In medieval ages belonged to Arnedo until 1669. In that year Grávalos become an independent municipality, but still remained as a possession of the Arnedo's Church. The municipality covers an area of 31.91 square kilometres and as of 2011 had a population of 234 people.</p>	<input type="range"/>	<input type="radio"/> yes <input type="radio"/> no
 <p>New Orleans New Orleans is a major United States port and the largest city and metropolitan area in the state of Louisiana. The population of the city was 343,829 as of the 2010 U.S. Census. The New Orleans metropolitan area (New Orleans–Metairie–Kenner Metropolitan Statistical Area) had a population of 1,167,764 in 2010 and was the 46th largest in the United States. The New Orleans–Metairie–Bogalusa Combined Statistical Area, a larger trading area, had a 2010 population of 1,214,932.</p>	<input type="range"/>	<input type="radio"/> yes <input type="radio"/> no
 <p>Albinyana Albinyana (official name in Catalán) or Albiñana is a village in the province of Tarragona and autonomous community of Catalonia, Spain. It belongs to Tarragona in the Baix Penedès region. According to data from 2009, its population was at 2,275 inhabitants.</p>	<input type="range"/>	<input type="radio"/> yes <input type="radio"/> no

Fig. B.19: Travel - Results part II, TC2 (page 6)

 <p>Sardinia Sardinia is the second largest island in the Mediterranean Sea and an autonomous region of Italy. The nearest land masses are (clockwise from north) the island of Corsica, the Italian Peninsula, Sicily, Tunisia and the Balearic Islands.</p>	<input type="checkbox"/> yes <input type="checkbox"/> no
 <p>Sicily Sicily is the largest island in the Mediterranean Sea; along with surrounding minor islands, it constitutes an autonomous region of Italy, the Regione Siciliana (Sicilian Region). Sicily is located in the central Mediterranean. It extends from the tip of the Apennine peninsula from which it is separated only by the narrow Strait of Messina, towards the North African coast.</p>	<input type="checkbox"/> yes <input type="checkbox"/> no
 <p>Azores The Archipelago of the Azores is composed of nine volcanic islands situated in the North Atlantic Ocean, and is located about 1,500 km west of Lisbon and about 1,900 km southeast of Newfoundland. The islands, and their Exclusive Economic Zone, form the Autonomous Region of the Azores, one of the two autonomous regions of Portugal.</p>	<input type="checkbox"/> yes <input type="checkbox"/> no
 <p>Madeira Madeira is a Portuguese archipelago that lies between <code>{{#invoke:Coordinates coord}} {{#coordinates:32 22.3 N 16 16.5 W name= }}</code> and <code>{{#invoke:Coordinates coord}} {{#coordinates:33 7.8 N 17 16.65 W name= }}</code>, just under 400 kilometres north of Tenerife, Canary Islands, in the north Atlantic Ocean and an outermost region of the European Union.</p>	<input type="checkbox"/> yes <input type="checkbox"/> no
 <p>Balearic Islands The Balearic Islands are an archipelago of Spain in the western Mediterranean Sea, near the eastern coast of the Iberian Peninsula. The four largest islands are: Majorca, Minorca, Ibiza and Formentera. The archipelago forms an autonomous community and a province of Spain, with Palma as the capital city. The co-official languages in the Balearic Islands are Catalan and Spanish. The current Statute of Autonomy declares the Balearic Islands as one nationality of Spain.</p>	<input type="checkbox"/> yes <input type="checkbox"/> no

Statement	Agreement
The recommendations of list 2 better fit my preferences than the suggestions I would receive from sources that I usually use to find out about interesting places (see Q6).	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The recommendations of list 2 are diverse.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
I feel supported to find interesting travel destinations with the help of the recommendations of list 2.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The filter has improved the recommendations of list 2.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree

Fig. B.20: Travel - Results part III, TC2 (page 6)

Home Contact

Q10. Test Case III

In this section you will receive recommendations based on your preferences for certain places. Choose a city (e.g. Berlin), a region (e.g. Tuscany) or a country (e.g. Greece) and provide your favorite places there (e.g. Berlin Wall, Unter den Linden and Museum Island for Berlin). We kindly ask you to only select places that appear in the auto-complete lists as soon as you type.

Please note: When searching for words, mutated vowels should be replaced by interrogation marks.

Travel Destination
City: London ID: London IMG:

Favorite places there
Destination 1 Oxford Street ID: Oxford
Destination 2 Notting Hill ID: Notting_H
Destination 3 South Bank

Please note: **South Bank ferry wharf (Brisbane)**
South Bank. 1 & 2 is a ferry wharf in the suburb of South Brisbane used by the CityCat on the Brisbane River.

South Bank
The South Bank is an area of Central London, England located immediately adjacent to the south bank of the River Thames. It forms a long and narrow section of riverside development within the London Borough of Lambeth and the London Borough of Southwark. It developed much more slowly than the north bank of the river due to adverse conditions, and throughout its history has twice functioned as an entertainment district, separated by a hundred years of use as a location for industry.


South Bank Parklands
The South Bank Parklands are located at South Bank in Brisbane, Queensland, Australia. The parkland, on the transformed site of Brisbane's World Expo 88, was officially opened to

Fig. B.21: Travel - User profile generation, TC3 (page 7)


Q10. Results of Test Case III


Below, you find recommendations by different algorithms that fit your inputs in test case III. For each destination in one of the 2 recommendation lists, move the slider to evaluate its relevance (outer left side: lowest relevance, outer right side: highest relevance). In addition to that, tick one of the buttons in the outer right column to indicate, whether a destination is new to you or not. Please examine the overall usefulness of each recommendation list on the bottom of the list


Your favored place was...

 London

There you liked the following places of interest (POI)...

 Oxford Street

 Notting Hill

 South Bank

Recommendation List 1













Title	Relevant	New
 <p>Wallingford, Oxfordshire Wallingford is an ancient market town and civil parish in the upper Thames Valley in England. Historically in Berkshire, it was transferred to Oxfordshire in 1974. Wallingford is situated 12 miles (19 km) north of Reading, 13 miles (21 km) south of Oxford and 11 miles (18 km) north west of Henley-on-Thames. The town's royal but mostly ruined Wallingford Castle held high status in the early medieval period as a regular royal residence until the Black Death hit the town badly in 1349. Empress Matilda retreated here for the final time from Oxford Castle in 1141. The castle declined subsequently, much stone being removed to renovate Windsor Castle. Nonetheless the town's Priory produced two of the greatest minds of the age, the mathematician Richard of Wallingford and the chronicler John of Wa</p>		<input type="radio"/> yes <input type="radio"/> no
 <p>Abingdon-on-Thames Abingdon / æbɪŋdɒn/, also known as Abingdon on Thames or Abingdon-on-Thames, is a market town and civil parish in England. Historically, it was the county town of Berkshire, but it has been in the administrative county of Oxfordshire since 1974. The 2011 Census recorded the parish's population as 33,130. This is 2,504 more than in the 2001 Census total of 30,626, and represents just over 8% growth in the population.</p>		<input type="radio"/> yes <input type="radio"/> no

Fig. B.22: Travel - Results part I, TC3 (page 8)

 <p>Oslo Oslo (English pronunciation: /ˈɒzləʊ/, OZ-loh, Norwegian pronunciation: [ˈʊsˀlʊ] or, rarer [ˈʊsˀlʊ] or [ˈʊslʊ]) is the capital and the most populous city in Norway. Oslo constitutes both a county and a municipality. Founded in the year 1040, and established as a "kaupstad" or trading place in 1048 by King Harald III, the city was elevated to a bishopric in 1070 and a capital under Haakon V around 1300. Personal unions with Denmark from 1397 to 1523 and again from 1536 to 1814 and with Sweden from 1814 to 1905 reduced its influence. After being destroyed by a fire in 1624, the city was moved closer to Akershus Fortress during the reign of King Christian IV and renamed Christiania in his honour. It was established as a municipality (formannskapsdistrikt) on 1 January 1838. Fol</p>		<input type="radio"/> yes <input type="radio"/> no
 <p>Bristol Bristol (/ˈbrɪstəl/) is a city, unitary authority area and county in South West England with an estimated population of 449,300 in 2016. It is England's sixth and the United Kingdom's eighth most populous city, and the most populous city in Southern England after London. The city borders the Unitary Authority areas of North Somerset and South Gloucestershire, with the historic cities of Bath and Gloucester to the south-east and north-east, respectively.</p>		<input type="radio"/> yes <input type="radio"/> no
 <p>Dublin Dublin (/ˈdʌblɪn/, Irish: Baile Átha Cliath [ˈblʲiː ˈclʲiə]) is the capital and largest city of Ireland. Dublin is in the province of Leinster on Ireland's east coast, at the mouth of the River Liffey. The city has an urban area population of 1,345,402. The population of the Greater Dublin Area, as of 2016, was 1,904,806 people.</p>		<input type="radio"/> yes <input type="radio"/> no
 <p>Athens Athens (/ˈæθɪnz/; Modern Greek: Αθήνα, Athína Greek pronunciation: [aˈθina], Ancient Greek: Ἀθῆναι, Athēnai) is the capital and largest city of Greece. Athens dominates the Attica region and is one of the world's oldest cities, with its recorded history spanning over 3,400 years, and its earliest human presence starting somewhere between the 11th and 7th millennia BC. Classical Athens was a powerful city-state that emerged in conjunction with the seagoing development of the port of Piraeus, which had been a distinct city prior to its 5th century BC incorporation with Athens. A centre for the arts, learning and philosophy, home of Plato's Academy and Aristotle's Lyceum, it is widely referred to as the cradle of Western civilization and the birthplace of democracy, largely because of its cul</p>		<input type="radio"/> yes <input type="radio"/> no

Statement	Agreement
The recommendations of list 2 better fit my preferences than the suggestions I would receive from sources that I usually use to find out about interesting places (see Q6).	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The recommendations of list 2 are diverse.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
I feel supported to find interesting travel destinations with the help of the recommendations of list 2.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree

Fig. B.24: Travel - Results part III, TC3 (page 8)

Recommender Systems
User Studies

[Home](#) [Contact](#)

Many thanks for your participation!

Important instruction:

Please copy the following code and paste it into the field provided within your Clickworker task form.

Your Clickworker fee can not be credited without the input of this code!

Code: Q820FL40

In case you have any suggestions, you can send them to us by using this form or via [e-mail!](#)

Fig. B.25: Travel - Finish screen (page 9)

B.2.3 Multimedia Experiments (Music Domain)

Recommender Systems
User Studies

[Home](#) [Contact](#)

Media Recommender Systems - Music

Thank you for participating in our experiment on media recommender systems! Your help is very much appreciated. The study examines the usefulness of recommendation algorithms when searching for music acts. Before taking part in this experiment, please read the following consent form and click on the "I Agree" button at the bottom of the page if you understand the statements and freely consent to participate in the study.

Consent Form

This web-based experiment is designed to understand preferences for different recommender systems' algorithms in the music domain. The study is being conducted by Lisa Wenige (M.Sc.), research assistant at the Chair of Business Information Systems, Friedrich Schiller University Jena, Germany. No deception is involved, and the study involves no more than minimal risk to participants (i.e. the level of risk encountered in daily life).

Participation in the study typically takes 20 minutes and is strictly anonymous. Participants start by providing demographic information and answering a few questions regarding their habits when searching for music acts. Then, they will be asked to evaluate recommendation results in 4 different music search test cases. Sufficient instructions are provided for each test case.

All responses are treated as confidential, and in no case will responses from individual participants be identified. Rather, all data will be published in aggregate form only. Participants should be aware, however, that the experiment is not being run from a "secure" https server of the kind typically used to handle credit card transactions, so there is a small possibility that responses could be viewed by unauthorized third parties (e.g. computer hackers).

Many individuals from previously conducted similar experiments found it to be at least interesting, if not enjoyable, and no adverse reactions have been reported thus far. Participants from clickworker.com will receive a monetary compensation. Other visitors to this web site are welcome to complete the study, although they will receive no compensation. Participation is entirely voluntary. Participants are free to refuse to answer any question or withdraw from the experiment at any time. The decision about whether to take part in the study will not affect any future interactions of the participants with media content providers or their relations with the Friedrich Schiller University Jena.

Participants who have further questions about this study or their rights, or wish to issue a complaint or concern may [contact us](#) via phone or e-mail.

If you are 18 years of age or older, understand the statements above, and freely consent to participate in the study, click on the "I Agree" button to begin the experiment.

Fig. B.26: Music - Consent form (page 1)

Recommender Systems

User Studies

[Home](#) [Contact](#)

Media Recommender Systems - Music

Please start by answering a few general questions about yourself and your music search habits.

Demographic Information

Q1. Your age range

Q2. Your Gender

Music

Q3. What is your favorite genre?

Q4. How many hours do you spend listening to music per day (on average)?

Music Search

Q5. How do you hear/find out about interesting music acts?
 (Multiple responses are possible)

- Family and friends
- Media coverage (TV, magazines, newspapers)
- Media store
- Local library
- Online communities (e.g. Last.fm, Slacker)
- Onlineshops, Streaming websites (e.g. Amazon, Spotify)
- Others

Q6. Please complete the following sentence: Finding interesting music acts is ...
 very easy easy manageable hard very hard

Fig. B.27: Music - Demographics section (page 2)

Q7. Test Case 1

This section explores the utility of automatically generated recommendations for music acts. To get a better idea of your preferences, please provide 3 music acts whose music you particularly like. For the recommendation algorithms to work, we kindly ask you to only select singers/bands from the auto-complete lists that appear as soon as you type.

Favorite music acts

Music act 1

Music act 2

Music act 3

Bono
 Paul David Hewson (born 10 May 1960), known by his stage name Bono, is an Irish singer, musician, venture capitalist and humanitarian best known for being the main vocalist of the Dublin-based rock band U2. Bono was born and raised in Dublin, Ireland, and attended Mount Temple Comprehensive School where he met his future wife, Alison Stewart, and the future members of U2. Bono writes almost all U2 lyrics, often using political, social, and religious themes.

Sarah De Bono
 Sarah De Bono is an Australian singer-songwriter and pianist, born and raised in Melbourne. She participated on the first season of The Voice (Australia), coming in fourth place. Shortly after she signed a record deal with Universal Music Australia. On 24 June 2012, De Bono scored her first top 10 hit with "Beautiful", which peaked at number four on the ARIA Singles Chart and was certified gold.

Elijah Blue Allman
 Elijah Blue Allman (born July 10, 1976) is the son of Cher and her second husband Gregg Allman and half brother of Chaz Bono, Delliha Allman, Michael Allman, Layla Allman and Devon




Fig. B.28: Music - User profile generation, TC1 (page 3)

Q7. Results of Test Case I

Below, you find recommendations that are provided to you according to your favorite music acts in test case I. For each music act in the recommendation list, move the slider to evaluate its relevance (outer left side: lowest relevance, outer right side: highest relevance). In addition to that, tick one of the buttons in the outer right column to indicate, whether a music act is new to you or not. Please examine the overall usefulness of the recommendation list on the bottom of the page.

Please open hyperlinks in a new tab. Otherwise your evaluations are gone.

Your favorite music acts were...

-  [The Police](#)
-  [Sting \(musician\)](#)
-  [Bono](#)

Recommendations







Title	Relevant	New
 <p>Paul McCartney Sir James Paul McCartney, MBE (born 18 June 1942) is an English musician, singer-songwriter, multi-instrumentalist and composer. With John Lennon, George Harrison and Ringo Starr, he gained worldwide fame as a member of the Beatles, and his collaboration with Lennon is one of the most celebrated songwriting partnerships of the 20th century. After the band's break-up, he pursued a solo career, later forming Wings with his first wife, Linda, and singer-songwriter Denny Laine.</p>		<input type="radio"/> yes <input type="radio"/> no
 <p>David Bowie David Robert Jones (born 8 January 1947), known by his stage name David Bowie, is an English musician, singer-songwriter, record producer, actor, and arranger. Bowie has been a major figure in the world of popular music for over four decades, and is renowned as an innovator, particularly for his work in the 1970s. He is known for his distinctive voice as well as the intellectual depth and eclecticism of his work.</p>		<input type="radio"/> yes <input type="radio"/> no
 <p>John Lennon John Ono Lennon, MBE (born John Winston Lennon; 9 October 1940 – 8 December 1980) was an English musician, singer and songwriter who rose to worldwide fame as a founder member of the Beatles, one of the most commercially successful and critically acclaimed acts in the history of popular music. With Paul McCartney, he formed one of the most celebrated songwriting partnerships of the 20th century.</p>		<input type="radio"/> yes <input type="radio"/> no

Fig. B.29: Music - Results part I, TC1 (page 4)

	<p>Eric Clapton Eric Patrick Clapton, CBE, (born 30 March 1945) is an English musician, singer, songwriter and guitarist. He is the only three-time inductee to the Rock and Roll Hall of Fame: once as a solo artist, and separately as a member of The Yardbirds and Cream. Clapton has been referred to as one of the most important and influential guitarists of all time.</p>	<input type="range"/>	<input type="radio"/> yes <input type="radio"/> no
	<p>Robin Gibb Robin Hugh Gibb, CBE (22 December 1949 – 20 May 2012) was a singer and songwriter, best known as a member of the Bee Gees, co-founded with his fraternal twin brother Maurice and older brother Barry. Their younger brother Andy was also a singer. Born in the Isle of Man to English parents, the family later moved to Manchester before settling in Redcliffe, a suburb of Brisbane, Australia.</p>	<input type="range"/>	<input type="radio"/> yes <input type="radio"/> no
	<p>Elton John Sir Elton Hercules John, CBE (born Reginald Kenneth Dwight on 25 March 1947), is an English rock singer-songwriter, composer, pianist and occasional actor. He has worked with lyricist Bernie Taupin as his songwriter partner since 1967; they have collaborated on more than 30 albums to date. In his four-decade career John has sold more than 250 million records, making him one of the most successful artists of all time.</p>	<input type="range"/>	<input type="radio"/> yes <input type="radio"/> no
	<p>Robert Plant Robert Anthony Plant CBE (born 20 August 1948) is an English musician, singer and songwriter. Best known as the lead vocalist and lyricist of the rock band Led Zeppelin, he also had a successful solo career. With a career spanning more than 40 years, Plant is regarded as one of the most significant singers in the history of rock music, and has influenced contemporaries and later singers such as Freddie Mercury, Axl Rose and Chris Cornell.</p>	<input type="range"/>	<input type="radio"/> yes <input type="radio"/> no


Statement	Agreement
The recommendations better fit my preferences than the suggestions I would receive from other sources (see Q5).	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
I feel supported to find interesting music acts with the help of the recommender.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The recommended items influence my selection of music acts.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The recommended items effectively helped me to find what I like.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
I feel supported in selecting the items to buy with the help of the recommender.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The items recommended to me are diverse.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The items recommended to me are similar to each other.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree


Fig. B.30: Music - Results part II, TC1 (page 4)


Q8. Test Case II

In this test case you can filter recommendations according to your interests. You see the music acts you have stated as your preferences in the last section (Q7). Now, you are able to refine these preferences with a condition. Please choose either a subject (e.g. Folk singers) or a genre (e.g. Soul music) that fits your interests. For the recommendation algorithms to work, we kindly ask you to only select conditions that appear in the auto-complete list of the "Filter" field.

Favorite music acts

 The Police

 Sting (musician)

 Bono

Filter

Subject

- English guitarists
- English songwriters
- English toponyms
- English composers
- English musicians
- English rappers
- English folklore
- English contraltos




Fig. B.31: Music - User profile generation, TC2 (page 5)

Q8. Results of Test Case II

Below, you find recommendations that are provided to you according to your favorite music acts and your filter condition. For each music act in the two recommendation lists, move the slider to evaluate its relevance (outer left side: lowest relevance, outer right side: highest relevance). In addition to that, tick one of the buttons in the outer right column to indicate, whether a music act is new to you or not. Please examine the overall usefulness of the recommendation list on the bottom of each list.

Please open hyperlinks in a new tab. Otherwise your evaluations are gone.

Your favorite music acts were...

-  [The Police](#)
-  [Sting \(musician\)](#)
-  [Bono](#)

Your condition was: **English musicians**

Recommendation List 1







Title	Relevant	New
 <p>Fuzz (musician) James Robert Lombard, OBE (born 9 February 1952), professionally known by his stage name Fuzz, is an English singer-songwriter, musician, actor, record producer and composer noted for being the frontman of the band Inferno. Fuzz began his career in 1969 at the age of 17, when he formed Inferno alongside MacMick, Leon O'Brien and James Coolidge.</p>		<input type="radio"/> yes <input type="radio"/> no
 <p>Dizze Rascal Dylan Kwabena Mills (born 1 October 1985), better known by his stage name Dizzee Rascal, is an English rapper, MC, songwriter and record producer with African roots in Ghana and Nigeria. His music is a blend of grime, garage, bassline, hip hop, rap, and R&B. Best known for his number-one hits "Dance wiv Me," "Bonkers," "Holiday," "Dirtee Disco" & "Shout," his debut album, <i>Boy in da Corner</i>, won him the 2003 Mercury Prize.</p>		<input type="radio"/> yes <input type="radio"/> no
 <p>Steven Reid Williams Steven Reid Williams (born March 1976) is an English singer-songwriter born in Bristol, England.</p>		<input type="radio"/> yes <input type="radio"/> no

Fig. B.32: Music - Results part I, TC2 (page 6)

Statement	Agreement
The recommendations better fit my preferences than the suggestions I would receive from other sources (see Q5).	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
I feel supported to find interesting music acts with the help of the recommender.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The recommended items influence my selection of music acts.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The recommended items effectively helped me to find what I like.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
I feel supported in selecting the items to buy with the help of the recommender.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The items recommended to me are diverse.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The items recommended to me are similar to each other.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The filter has improved the recommendations of list 1.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree

Recommendation List 2










Title	Relevant	New
 <p>Elton John Sir Elton Hercules John, CBE (born Reginald Kenneth Dwight on 25 March 1947), is an English rock singer-songwriter, composer, pianist and occasional actor. He has worked with lyricist Bernie Taupin as his songwriter partner since 1967; they have collaborated on more than 30 albums to date. In his four-decade career John has sold more than 250 million records, making him one of the most successful artists of all time.</p>	<input type="range"/>	<input type="radio"/> yes <input type="radio"/> no
 <p>Robert Plant Robert Anthony Plant CBE (born 20 August 1948) is an English musician, singer and songwriter. Best known as the lead vocalist and lyricist of the rock band Led Zeppelin, he also had a successful solo career. With a career spanning more than 40 years, Plant is regarded as one of the most significant singers in the history of rock music, and has influenced contemporaries and later singers such as Freddie Mercury, Axl Rose and Chris Cornell.</p>	<input type="range"/>	<input type="radio"/> yes <input type="radio"/> no
 <p>Jimmy Page James Patrick "Jimmy" Page OBE (born 9 January 1944) is the English musician, songwriter and record producer who achieved international success as the guitarist of the rock band Led Zeppelin. Page began his career as a studio session musician in London and by the mid-1960s, had become the most sought-after session guitarist in the UK. He was a member of the Yardbirds from 1966 to 1968. In late 1968, he founded Led Zeppelin.</p>	<input type="range"/>	<input type="radio"/> yes <input type="radio"/> no

Fig. B.33: Music - Results part II, TC2 (page 6)

 <p>David Gilmour David Jon Gilmour, CBE (born 6 March 1946) is an English musician and multi-instrumentalist, who was the guitarist, lead vocalist and songwriter of the progressive rock band Pink Floyd. It is estimated that as of 2010, the group have sold over 250 million records worldwide, including 74.5 million units sold in the United States. In addition to his work with Pink Floyd, Gilmour has worked as a producer for a variety of artists, and has enjoyed a successful career as a solo artist.</p>		<input type="radio"/> yes <input type="radio"/> no
 <p>Freddie Mercury Freddie Mercury; 5 September 1946 – 24 November 1991 was a British musician, singer and songwriter, best known as the lead vocalist and lyricist of the rock band Queen. As a performer, he was known for his flamboyant stage persona and powerful vocals over a four-octave range. As a songwriter, Mercury composed many hits for Queen, including "Bohemian Rhapsody", "Killer Queen", "Somebody to Love", "Don't Stop Me Now", "Crazy Little Thing Called Love" and "We Are the Champions".</p>		<input type="radio"/> yes <input type="radio"/> no
 <p>Damon Albarn Damon Albarn is an English musician, singer-songwriter, record producer and actor who came to prominence as the frontman and primary songwriter of the alternative rock band Blur. However, he has helmed many other high profile projects, most notably Gorillaz, a virtual band. Raised in Leytonstone, London and around Colchester, Essex, Albarn started learning guitar, piano and violin in his youth.</p>		<input type="radio"/> yes <input type="radio"/> no

Statement	Agreement
The recommendations better fit my preferences than the suggestions I would receive from other sources (see Q5).	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
I feel supported to find interesting music acts with the help of the recommender.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The recommended items influence my selection of music acts.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The recommended items effectively helped me to find what I like.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
I feel supported in selecting the items to buy with the help of the recommender.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The items recommended to me are diverse.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The items recommended to me are similar to each other.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The filter has improved the recommendations of list 2.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree

Fig. B.34: Music - Results part III, TC2 (page 6)

Q9. Test Case III

In this test case you can obtain recommendations based on other algorithms. Please select a music act you particularly like. For the recommendation algorithms to work, we kindly ask you to only select items that appear in the auto-complete list of the text field.

Favorite music act

Sting

Strontium 90 (band)
Strontium 90 was the name of a short-lived 1977 British band with members Mike Howlett (lead bass, vocals), Sting (bass, vocals), Stewart Copeland (drums), and Andy Summers (guitar). The band is most notable for introducing Summers to Sting and Copeland, as this trio would go on to massive success as The Police. The band was formed in mid-1977 by Howlett after he quit Gong and recruited Sting and Summers to participate in a new project.

Sting (musician)
Gordon Matthew Thomas Sumner CBE (born 2 October 1951), known by his stage name Sting, is an English musician, singer-songwriter, multi-instrumentalist, activist, actor and philanthropist. He was the principal songwriter, lead singer and bassist for the rock band The Police before launching a solo career. Sting has varied his musical style, incorporating distinct elements of jazz, reggae, classical, New Age, and worldbeat into his music.

Dominic Miller
Dominic Miller (born 21 March 1960) is an Argentine-English guitarist who toured and recorded with World Party and King Swamp, worked on Phil Collins' solo album ... But Seriously and played

Fig. B.35: Music - User profile generation, TC3 (page 7)

Q9. Results of Test Case III

Below, you find recommendations that are provided to you according to your favorite music act. For each recommendation in the two lists, move the slider to evaluate its relevance (outer left side: lowest relevance, outer right side: highest relevance). In addition to that, tick one of the buttons in the outer right column to indicate, whether an item is new to you or not. Please examine the overall usefulness of the recommendation list on the bottom of each list.

Please open hyperlinks in a new tab. Otherwise your evaluations are gone.

Please note: In case one of the recommendation lists is empty, just skip the evaluation for the respective list.

Your favorite music act was...



Recommendation List 1







Title	Relevant	New
 <p>Phil Collins Philip David Charles "Phil" Collins, LVO (born 30 January 1951) is an English singer-songwriter, multi-instrumentalist and actor best known as a drummer and vocalist for English progressive rock group Genesis and as a solo artist. Collins sang the lead vocals on several chart hits in the United Kingdom and the United States between 1976 and 2010, either as a solo artist or with Genesis.</p>		<input type="radio"/> yes <input type="radio"/> no
 <p>John Lennon John Ono Lennon, MBE (born John Winston Lennon; 9 October 1940 – 8 December 1980) was an English musician, singer and songwriter who rose to worldwide fame as a founder member of the Beatles, one of the most commercially successful and critically acclaimed acts in the history of popular music. With Paul McCartney, he formed one of the most celebrated songwriting partnerships of the 20th century.</p>		<input type="radio"/> yes <input type="radio"/> no
 <p>David Bowie David Robert Jones (born 8 January 1947), known by his stage name David Bowie, is an English musician, singer-songwriter, record producer, actor, and arranger. Bowie has been a major figure in the world of popular music for over four decades, and is renowned as an innovator, particularly for his work in the 1970s. He is known for his distinctive voice as well as the intellectual depth and eclecticism of his work.</p>		<input type="radio"/> yes <input type="radio"/> no

Fig. B.36: Music - Results part I, TC3 (page 8)

Statement	Agreement
The recommendations better fit my preferences than the suggestions I would receive from other sources (see Q5).	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
I feel supported to find interesting music genres and music acts with the help of the recommender.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The recommended items influence my selection.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The recommended items effectively helped me to find what I like.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
I feel supported in selecting music genres and music acts to listen to with the help of the recommender.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The items recommended to me are diverse.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The items recommended to me are similar to each other.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree

Recommendation List 2












Title	Relevant	New
 <p>Peter Frampton Peter Kenneth Frampton (born 22 April 1950) is an English musician, singer, songwriter, producer, guitarist and multi-instrumentalist. He was previously associated with the bands Humble Pie and The Herd. Frampton's international breakthrough album was his live release, Frampton Comes Alive!. The album sold more than six million copies in the United States alone and spawned several hits. Since then he has released several major albums.</p>	<input type="range"/>	<input type="radio"/> yes <input type="radio"/> no
 <p>Squeeze (band) Squeeze are a British band that came to prominence in the United Kingdom during the New Wave period of the late 1970s and continued recording successfully in the 1980s and 1990s. They are known in the UK for their hit songs "Cool for Cats", "Up the Junction", "Tempted", "Labelled With Love", "Black Coffee In Bed", "Another Nail in My Heart", "Pulling Mussels (From the Shell)" and "Hourglass". Though not as commercially successful in the U.S.</p>	<input type="range"/>	<input type="radio"/> yes <input type="radio"/> no
 <p>Joan Armatrading Joan Anita Barbara Armatrading, MBE (born 9 December 1950) is a British singer, songwriter and guitarist. Armatrading is a three-time Grammy Award-nominee and has been nominated twice for BRIT Awards as Best Female Artist. She also received an Ivor Novello Award for Outstanding Contemporary Song Collection in 1996. In a recording career spanning 40 years she has released a total of 18 studio albums, as well as several live albums and compilations.</p>	<input type="range"/>	<input type="radio"/> yes <input type="radio"/> no

Fig. B.37: Music - Results part II, TC3 (page 8)

 <p>Nazareth (band) Nazareth are a Scottish hard rock band, founded in 1968, that had several hits in the United Kingdom in the early 1970s, and established an international audience with their 1975 album Hair of the Dog. Perhaps their best-known hit single was a cover of the ballad "Love Hurts", in 1975. The band continue to record and tour.</p>		<input type="radio"/> yes <input type="radio"/> no
 <p>Cat Stevens Yusuf Islam (born Steven Demetre Georgiou; 21 July 1948), commonly known by his former stage name Cat Stevens, is a British singer-songwriter, multi-instrumentalist, humanitarian, education philanthropist, and prominent convert to Islam. His early 1970s record albums Tea for the Tillerman and Teaser and the Firecat were both certified triple platinum in the United States by the RIAA.</p>		<input type="radio"/> yes <input type="radio"/> no
 <p>Fairport Convention Fairport Convention are an English folk rock and electric folk band. Formed in 1967, they are widely regarded as the most important group in the English folk rock movement. Their seminal album Liege & Lief, is considered to have launched the electric folk or English folk rock movement, which provided a distinctively English identity to rock music and helped awaken much wider interest in traditional music in general.</p>		<input type="radio"/> yes <input type="radio"/> no
 <p>Supertramp Supertramp are a British rock band formed in 1969 under the name Daddy before renaming themselves in early 1970. Though their music was initially categorised as progressive rock, they have since incorporated a combination of traditional rock, pop and art rock into their music. The band's work is marked by the inventive songwriting of Rick Davies and Roger Hodgson, the distinctive voice of Hodgson, and the prominent use of Wuritzer electric piano and saxophone in their songs.</p>		<input type="radio"/> yes <input type="radio"/> no

Statement	Agreement
The recommendations better fit my preferences than the suggestions I would receive from other sources (see Q5).	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
I feel supported to find interesting music genres and music acts with the help of the recommender.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The recommended items influence my selection.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The recommended items effectively helped me to find what I like.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
I feel supported in selecting the music genres and music acts to listen to with the help of the recommender.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The items recommended to me are diverse.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The items recommended to me are similar to each other.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree

Fig. B.38: Music - Results part III, TC3 (page 8)

Q10. Test Case IV

This section explores cross-domain recommendations. Please provide a music act you particularly like. Then, the system identifies movies, that fit your input. For the recommendation algorithms to work, we kindly ask you to only select items that appear in the auto-complete list of the text field.

A music act you want to receive movie recommendations for ...

Bono

Bono
Paul David Hewson (born 10 May 1960), known by his stage name Bono, is an Irish singer, musician, venture capitalist and humanitarian best known for being the main vocalist of the Dublin-based rock band U2. Bono was born and raised in Dublin, Ireland, and attended Mount Temple Comprehensive School where he met his future wife, Alison Stewart, and the future members of U2. Bono writes almost all U2 lyrics, often using political, social, and religious themes.

Sarah De Bono

Sarah De Bono is an Australian singer-songwriter and pianist, born and raised in Melbourne. She participated on the first season of The Voice (Australia), coming in fourth place. Shortly after she signed a record deal with Universal Music Australia. On 24 June 2012, De Bono scored her first top 10 hit with "Beautiful", which peaked at number four on the ARIA Singles Chart and was certified gold.

Elijah Blue Allman

Elijah Blue Allman (born July 10, 1976) is the son of Cher and her second husband Gregg Allman and half brother of Chaz Bono, Dallah Allman, Michael Allman, Layla Allman and Devon Allman.

Jon Levine

Jon Levine is a Canadian producer, songwriter, and former keyboardist and songwriter for The Philosopher Kings. He has written and produced songs for artists such as Nelly Furtado, Cher Lloyd, T-Boz, K'Neam, Bono, Selena Gomez.

Fig. B.39: Music - User profile generation, TC4 (page 9)


Q10. Results of Test Case IV

Below, you find recommendations for movies that are provided to you according to your stated preference for a music act. For each item in the two recommendation lists, move the slider to evaluate its relevance (outer left side: lowest relevance, outer right side: highest relevance). In addition to that, tick one of the buttons in the outer right column to indicate, whether an item is new to you or not. Please examine the overall usefulness of the recommendation list on the bottom of each list.

Please open hyperlinks in a new tab. Otherwise your evaluations are gone.

Please note: In case one of the recommendation lists is empty, just skip the evaluation for the respective list.

Your preferred music act was...





Bono

Recommendation List 1

Title	Relevant	New
Leonard Cohen: I'm Your Man Leonard Cohen: I'm Your Man is a 2005 film by Lian Lunson about the life and career of Leonard Cohen. It is based on a January 2005 tribute show at the Sydney Opera House titled "Came So Far for Beauty", which was produced by Hal Willner.		<input type="radio"/> yes <input type="radio"/> no
GoldenEye GoldenEye (1995) is the seventeenth spy film in the James Bond series, and the first to star Pierce Brosnan as the fictional MI6 officer James Bond. The film was directed by Martin Campbell and is the first film in the series not to take story elements from the works of novelist Ian Fleming. The story was conceived and written by Michael France, with later collaboration by other writers.		<input type="radio"/> yes <input type="radio"/> no
No Maps for These Territories No Maps for These Territories is an independent documentary film made by Mark Neale focusing on the speculative fiction author William Gibson. It features appearances by Jack Womack, Bruce Sterling, Bono, and The Edge and was released by Docurama. The film had its world premiere at the Vancouver International Film Festival in October 2000.		<input type="radio"/> yes <input type="radio"/> no
Rewind (video) Rewind is a DVD by Welsh rock trio Stereophonics released in 2007. It contains over three hours of live and documentary footage spanning their entire career, from pre-Stereophonics years to their signing to V2 Records in 1996 to 2006. A booklet was also included with the DVD, featuring several previously unseen photographs of the band, from Kelly Jones' own personal photo album.		<input type="radio"/> yes <input type="radio"/> no
From the Sky Down From the Sky Down is a 2011 documentary film directed by Davis Guggenheim about rock band U2 and the production of their 1991 album Achtung Baby. The film documents the album's difficult recording period, the band members' relationships, and the group's creative process. Guggenheim, who was commissioned by U2 to create the film to commemorate the record's 20th anniversary, spent several months in 2011 developing the documentary.		<input type="radio"/> yes <input type="radio"/> no

Fig. B.40: Music - Results part I, TC4 (page 10)

<p>White Diamond: A Personal Portrait of Kylie Minogue White Diamond: A Personal Portrait of Kylie Minogue is a 2007 documentary film directed and produced by William Baker and chronicling the life of Australian singer Kylie Minogue during her concert tour Showgirl: The Homecoming Tour. It was filmed between August 2006 and March 2007 in both Australia and the United Kingdom.</p>		<input type="radio"/> yes <input type="radio"/> no
<p>Being Mick Being Mick is a 2001 television film which chronicles the life of Mick Jagger for one year. Much of the film was filmed by Mick using a handheld camera. The film documents his recording of the Goddess in the Doorway album, as well as daily life including his family and friends. In the film, Mick attends a charity fundraiser hosted by Elton John as well as the premiere of the Kate Winslet film Enigma, which Jagger's company produced.</p>		<input type="radio"/> yes <input type="radio"/> no

Statement	Agreement
The recommendations better fit my preferences than the suggestions I would receive from other sources.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
I feel supported to find interesting movies with the help of the recommender.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The recommended items influence my selection.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The recommended items effectively helped me to find what I like.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
I feel supported in selecting the movies to watch with the help of the recommender.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The items recommended to me are diverse.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The items recommended to me are similar to each other.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree

Recommendation List 2

Title	Relevant	New

Statement	Agreement
The recommendations better fit my preferences than the suggestions I would receive from other sources.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
I feel supported to find interesting movies with the help of the recommender.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The recommended items influence my selection.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The recommended items effectively helped me to find what I like.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
I feel supported in selecting the movies to buy with the help of the recommender.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The items recommended to me are diverse.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree
The items recommended to me are similar to each other.	<input type="radio"/> strongly agree <input type="radio"/> agree <input type="radio"/> neutral <input type="radio"/> disagree <input type="radio"/> strongly disagree

Fig. B.41: Music - Results part II, TC4 (page 10)

Recommender Systems
User Studies

[Home](#) [Contact](#)

Many thanks for your participation!

Important instruction:

Please copy the following code and paste it into the field provided within your Clickworker task form.

Your Clickworker fee can not be credited without the input of this code!

Code: ML48ST94

In case you have any suggestions, you can send them to us by using this form or via e-mail!

Fig. B.42: Music - Finish screen (page 11)

B.3 Correlation Matrices

Table B.1: Spearman's ρ for evaluation metrics correlation with *perceived usefulness*, TC2

Domain	Method	Accuracy				Nov.	Diversity	
		<i>size</i>	<i>mrs</i>	<i>prec</i>	<i>ndcg</i>	<i>mv</i>	<i>divU</i>	<i>divC</i>
DL	Constr.-based (Regular)	-	-	0.43*	-	-	0.58**	-
Travel	Constr.-based (Regular)	0.49*	-	-	-	-	-	-
	Constr.-based (Expanded)	-	0.40**	-	-	-	0.44***	-
Movie	Constr.-based (Regular)	-	0.46**	0.42*	-	-	-	-
	Constr.-based (Expanded)	-	0.55***	0.48***	-	-	-	-
Music	Constr.-based (Regular)	0.50**	-	-	-	-	0.44*	-
	Constr.-based (Expanded)	0.35*	0.40**	-	-	-	0.35*	-
Book	Constr.-based (Regular)	-	-	-	-	-	-	-
	Constr.-based (Expanded)	-	-	-	-	-	-	-

Table B.2: Spearman's ρ for dependencies among evaluation metrics, TC2

Domain	Method	Accuracy			Diversity
		$\rho_{mrs,prec}$	$\rho_{mrs,ndcg}$	$\rho_{prec,ndcg}$	$\rho_{divU,divC}$
DL	Constr.-based (Regular)	0.82***	0.46*	-	-
Travel	Constr.-based (Regular)	0.55*	-	0.45*	-
	Constr.-based (Expanded)	0.71***	0.61***	0.76***	-
Movie	Constr.-based (Regular)	0.64***	-	0.47**	-
	Constr.-based (Expanded)	0.63***	-	0.53***	-
Music	Constr.-based (Regular)	-	-	0.56***	-
	Constr.-based (Expanded)	0.47**	-	0.57***	-
Book	Constr.-based (Regular)	0.63***	-	0.50**	-
	Constr.-based (Expanded)	0.66***	0.32*	0.60***	-

Table B.3: Spearman's ρ for evaluation metrics correlation with *perceived usefulness*, TC3

Domain	Method	Accuracy				Nov.	Diversity	
		<i>size</i>	<i>mrs</i>	<i>prec</i>	<i>ndcg</i>	<i>nv</i>	<i>divU</i>	<i>divC</i>
Travel	Regular	NA	0.62***	0.51***	0.26**	0.32**	0.77***	-
	Rollup	-	0.36***	0.30**	-	0.26**	0.69**	-
Movie	Regular	NA	-	-	-	-	-	-
	Rollup	NA	0.49***	0.47**	-	-	0.37*	-0.35*
Music	Regular	NA	0.53***	0.32*	-	-	-	-
	Rollup	NA	0.63***	-	-	-	-	-
Book	Regular	-	-	-	-	-	-	-
	Rollup	NA	0.63***	0.47***	-	-	0.32*	-

Table B.4: Spearman's ρ for dependencies among evaluation metrics, TC3

Domain	Method	Accuracy			Diversity
		$\rho_{mrs,prec}$	$\rho_{mrs,ndcg}$	$\rho_{prec,ndcg}$	$\rho_{divU,divC}$
Travel	Regular	0.86***	0.50***	0.66***	-
	Rollup	0.87***	0.59***	0.71***	-
Movie	Regular	0.75***	-	-	-
	Rollup	0.76***	-	0.36*	-
Music	Regular	0.56	-	0.63	-
	Rollup	0.47***	0.36**	0.77***	-
Book	Regular	0.68***	-	0.46**	-
	Rollup	0.72***	-	0.47**	-

Table B.5: Spearman's ρ for evaluation metrics correlation with *perceived usefulness*, TC4

Domain	Method	Accuracy				Nov.	Diversity	
		<i>size</i>	<i>mrs</i>	<i>prec</i>	<i>ndcg</i>	<i>nv</i>	<i>divU</i>	<i>divC</i>
Movie	Regular (SPARQL)	-	-	-	-	-	-	-
	Cross-Domain	-	0.50***	0.35*	-	-	-	-
Music	Regular (SPARQL)	0.45*	0.63**	0.68***	-	-	0.47*	-
	Cross-Domain	-	0.39*	-	-	-	-	-
Book	Regular (SPARQL)	-	-	0.47*	-	-	0.56**	-
	Cross-Domain	NA	0.60***	0.64***	0.49***	-	-	-

Table B.6: Spearman's ρ for dependencies among evaluation metrics, TC4

Domain	Method	Accuracy			Diversity
		$\rho_{mrs,prec}$	$\rho_{mrs,ndcg}$	$\rho_{prec,ndcg}$	$\rho_{divU,divC}$
Movie	Regular (SPARQL)	0.69**	-	0.75**	-
	Cross-Domain	0.69***	-	0.45**	-
Music	Regular (SPARQL)	0.74***	-	-	-
	Cross-Domain	0.65***	0.44**	0.80***	-
Book	Regular (SPARQL)	0.60**	-	0.56**	-
	Cross-Domain	0.64***	-	0.63***	-

Bibliography

- [1] RDF 1.1 concepts and abstract syntax. In: *W3C Recommendation* (2014)
- [2] *About data.bnf.fr*. URL: <http://data.bnf.fr/about>. 2018. – Accessed: January 2018
- [3] *About OpenLink Virtuoso*. URL: <https://virtuoso.openlinksw.com>. 2015. – Accessed: December 2016
- [4] ADOMAVICIUS, Gediminas ; TUZHILIN, Alexander: Multidimensional recommender systems: a data warehousing approach. In: *Proceedings of the 2nd International Workshop on Electronic Commerce (WELCOM'01)*, 2001, pp. 180–192
- [5] ADOMAVICIUS, Gediminas ; TUZHILIN, Alexander: Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. In: *IEEE Transactions on Knowledge and Data Engineering* 17 (2005), no. 6, pp. 734–749
- [6] ADOMAVICIUS, Gediminas ; TUZHILIN, Alexander ; ZHENG, Rong: REQUEST: a query language for customizing recommendations. In: *Information Systems Research* 22 (2011), no. 1, pp. 99–117
- [7] AGGARWAL, Charu C.: *Recommender systems: the textbook*. Springer, 2016
- [8] *AGRIS dataset description*. URL: <http://agris.fao.org/void.ttl>. 2017. – Accessed: February 2017
- [9] *AGROVOC*. URL: <http://aims.fao.org/aos/agrovoc/void.ttl>. – Accessed: February 2017
- [10] AHN, Jae-wook ; AMATRIAIN, Xavier: Towards fully distributed and privacy-preserving recommendations via expert collaborative filtering and restful Linked Data. In: *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)* vol. 1, 2010, pp. 66–73
- [11] *Amazon*. URL: <https://www.amazon.com>. 2018. – Accessed: March 2018
- [12] ANIBALDI, Stefano ; JAQUES, Yves. ; CELLI, Fabrizio ; STELLATO, Armando ; KEIZER, Johannes: Migrating bibliographic datasets to the Semantic Web: The AGRIS case. In: *Semantic Web* 6 (2015), no. 2, pp. 113–120

- [13] ANTELL, Karen ; HUANG, Jie: Subject searching success: Transaction logs, patron perceptions, and implications for library instruction. In: *Reference and User Services Quarterly* (2008), pp. 68–76
- [14] *Apache Jena - a free and open source Java framework for building Semantic Web and Linked Data applications*. URL: <https://jena.apache.org/>. 2017. – Accessed: December 2017
- [15] *Apache Solr 7.3.0*. URL: <http://lucene.apache.org/solr>. 2017. – Accessed: November 2017
- [16] *Apache Tomcat*. URL: <http://tomcat.apache.org>. – Accessed: December 2017
- [17] ARENAS, Marcelo ; CUENCA GRAU, Bernardo ; KHARLAMOV, Evgeny ; MARCIUSKA, Sarunas ; ZHELEZNYAKOV, Dmitriy: Faceted search over ontology-enhanced RDF data. In: *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management*, 2014, pp. 939–948
- [18] *ARQ - a SPARQL processor for Jena*. URL: <https://jena.apache.org/documentation/query>. 2017. – Accessed: December 2017
- [19] *arXiv.org*. URL: <https://arxiv.org>. 2017. – Accessed: June 2017
- [20] *Attribution 4.0 International (CC BY 4.0)*. URL: <https://creativecommons.org/licenses/by/4.0/deed>. – Accessed: February 2017
- [21] *Attribution-ShareAlike 3.0 Germany (CC BY-SA 3.0 DE)*. URL: <https://creativecommons.org/licenses/by-sa/3.0/de/deed.en>. – Accessed: July 2017
- [22] AUER, Sören ; BIZER, Christian ; KOBILAROV, Georgi ; LEHMANN, Jens ; CYGANIAK, Richard ; IVES, Zachary: DBpedia: a nucleus for a web of open data. In: *The Semantic Web*. Springer, 2007, pp. 722–735
- [23] AYALA, Victor A. A. ; PRZYJACIEL-ZABLOCKI, Martin ; HORNUNG, Thomas ; SCHÄTZLE, Alexander ; LAUSEN, Georg: Extending SPARQL for recommendations. In: *Proceedings of Semantic Web Information Management*, 2014, pp. 1–8
- [24] BAKER, Thomas ; BECHHOFFER, Sean ; ISAAC, Antoine ; MILES, Alistair ; SCHREIBER, Guus ; SUMMERS, Ed: Key choices in the design of Simple Knowledge Organization System (SKOS). In: *Web Semantics: Science, Services and Agents on the World Wide Web 20* (2013), pp. 35–49

- [25] BALLARD, Terry ; BLAINE, Anna: User search-limiting behavior in online catalogs: Comparing classic catalog use to search behavior in next-generation catalogs. In: *New Library World* 112 (2011), no. 5/6, pp. 261–273
- [26] BANNWART, Tobias ; BOUZA, Amancio ; REIF, Gerald ; BERNSTEIN, Abraham: Private cross-page movie recommendations with the Firefox add-on OMORE. In: *8th International Semantic Web Conference (ISWC 2009)*, 2009
- [27] BASILE, Pierpaolo ; MUSTO, Cataldo ; GEMMIS, Marco de ; LOPS, Pasquale ; NARDUCCI, Fedelucio ; SEMERARO, Giovanni: Aggregation strategies for Linked Open Data-enabled recommender systems. In: *11th European Semantic Web Conference (ESWC 2014)*, 2014
- [28] BECKER, Christian ; BIZER, Chris: *FlickrTM wrappr*. URL: <http://wifo5-03.informatik.uni-mannheim.de/flickrwrappr/>. – Accessed: April 2018
- [29] BEEL, Joeran ; BREITINGER, Corinna ; LANGER, Stefan ; LOMMATZSCH, Andreas ; GIPP, Bela: Towards reproducibility in recommender-systems research. In: *User Modeling and User-Adapted Interaction* 26 (2016), no. 1, pp. 69–101
- [30] BENDER, Michael A. ; FARACH-COLTON, Martin ; PEMMASANI, Giridhar ; SKIENA, Steveb ; SUMAZIN, Pavel: Lowest common ancestors in trees and directed acyclic graphs. In: *Journal of Algorithms* 57 (2005), no. 2, pp. 75–94
- [31] BENJAMINI, Yoav ; HOCHBERG, Yosef: Controlling the false discovery rate: a practical and powerful approach to multiple testing. In: *Journal of the Royal Statistical Society. Series B (Methodological)* 57 (1995), no. 1, pp. 289–300
- [32] BERNERS-LEE, Tim. *Linked Data*. URL: <https://www.w3.org/DesignIssues/LinkedData.html>
- [33] BERNERS-LEE, Tim ; CHEN, Yuhsin ; CHILTON, Lydia ; CONNOLLY, Dan ; DHANARAJ, Ruth ; HOLLENBACH, James ; LERER, Adam ; SHEETS, David: Tabulator: Exploring and analyzing Linked Data on the Semantic Web. In: *Proceedings of the 3rd International Semantic Web User Interaction Workshop*, 2006
- [34] BERNERS-LEE, Tim ; HENDLER, James ; LASSILA, Ora: The Semantic Web. In: *Scientific American* 284 (2001), no. 5, pp. 28–37
- [35] BILL, Robert W. ; LIU, Ying ; MCINNES, Bridget T. ; MELTON, Genevieve B.: Evaluating semantic relatedness and similarity measures with standardized MedDRA queries. In: *AMIA Annual Symposium Proceedings*, 2012, pp. 43–50
- [36] *Bio2RDF*. URL: <http://bio2rdf.org/>. – Accessed: April 2018

- [37] BIRKENBIHL, Klaus: Standards für das Semantic Web. In: PELLEGRINI, Tassilo (Eds.): *Semantic Web*. Springer, 2006, pp. 73–88
- [38] BIZER, Christian ; HEATH, Tom ; BERNERS-LEE, Tim: Linked Data: the story so far. In: *International Journal on Semantic Web and Information Systems* 5 (2009), no. 3, pp. 1–22
- [39] BIZER, Christian ; SCHULTZ, Andreas: The R2R framework: Publishing and discovering mappings on the web. In: *Proceedings of the 1st International Conference on Consuming Linked Data-Volume*, 2010, pp. 97–108
- [40] BOBADILLA, Jesús ; ORTEGA, Fernando ; HERNANDO, Antonio ; GUTIÉRREZ, Abraham: Recommender systems survey. In: *Knowledge-based Systems* 46 (2013), pp. 109–132
- [41] BORST, Timo: Repositorien auf ihrem Weg in das Semantic Web: semantisch hergeleitete Interoperabilität als Zielstellung für künftige Repository-Entwicklungen. In: *Bibliothek Forschung und Praxis* 38 (2014), no. 2, pp. 257–265
- [42] BORST, Timo: *Update: Neuer dump des EconStor LOD-Datensets steht zur Verfügung*. URL: <http://zbw.eu/labs/de/node/42>. 2016. – Accessed: February 2017
- [43] BOUTILIER, Craig ; BRAFMAN, Ronen I. ; DOMSHLAK, Carmel ; HOOS, Holger H. ; POOLE, David: CP-nets: a tool for representing and reasoning with conditional ceteris paribus preference statements. In: *Journal of Artificial Intelligence Research (JAIR)* 21 (2004), pp. 135–191
- [44] BUERLI, Mike ; OBISPO, CPSL: *The current state of graph databases*. URL: <https://pdfs.semanticscholar.org/5b5b/6b80badccd291e3437460222e24326c65979.pdf>. 2012. – Accessed: December 2016
- [45] BURKE, Robin: Hybrid recommender systems: Survey and experiments. In: *User Modeling and User-adapted Interaction* 12 (2002), no. 4, pp. 331–370
- [46] BUSHMAN, Barbara ; ANDERSON, David ; FU, Gang: Transforming the medical subject headings into Linked Data: creating the authorized version of MeSH in RDF. In: *Journal of Library Metadata* 15 (2015), no. 3/4, pp. 157–176
- [47] BÜTTCHER, Stefan ; CLARKE, Charles L. ; CORMACK, Gordon V.: *Information retrieval: Implementing and evaluating search engines*. MIT Press, 2016
- [48] CARACCILOLO, Caterina ; STELLATO, Armando ; MORSHED, Ahsan ; JOHANNSEN, Gudrun ; JAUQUES, Yves ; KEIZER, Johannes: Thesaurus maintenance, alignment and

- publication as Linked Data: the AGROVOC use case. In: *International Journal of Metadata, Semantics and Ontologies* 7 (2012), no. 1, pp. 65–75
- [49] CARPINETO, Claudio ; ROMANO, Giovanni: A survey of automatic query expansion in information retrieval. In: *ACM Computing Surveys (CSUR)* 44 (2012), no. 1
- [50] CASTELLS, Pablo ; VARGAS, Saül ; WANG, Jun: Novelty and diversity metrics for recommender systems: Choice, discovery and relevance. In: *International Workshop on Diversity in Document Retrieval (DDR 2011) at the 33rd European Conference on Information Retrieval (ECIR 2011)*, 2011
- [51] CELLI, Fabrizio ; MALAPELA, Thembani ; WEGNER, Karna ; SUBIRATS, Imma ; KOKOLIOU, Elena ; KEIZER, Johannes: AGRIS: Providing access to agricultural research data exploiting open data on the web. In: *F1000Research* 4 (2015)
- [52] CELMA, Òscar ; SERRA, Xavier: FOAFing the music: Bridging the semantic gap in music recommendation. In: *Web Semantics: Science, Services and Agents on the World Wide Web* 6 (2008), no. 4, pp. 250–256
- [53] CHENG, Gong ; GE, Weiyi ; QU, Yuzhong: Falcons: searching and browsing entities on the Semantic Web. In: *Proceedings of the 17th International Conference on World Wide Web*, 2008, pp. 1101–1102
- [54] *Clickworker - Datenmanagement für eCommerce*. URL: <https://www.clickworker.de/>. – Accessed: November 2017
- [55] CORBY, Olivier ; DIENG-KUNTZ, Rose ; FARON-ZUCKER, Catherine: Querying the Semantic Web with corese search engine. In: *Proceedings of the 16th European Conference on Artificial Intelligence*, 2004, pp. 705–709
- [56] CREMONESI, Paolo ; KOREN, Yehuda ; TURRIN, Roberto: Performance of recommender algorithms on top-n recommendation tasks. In: *Proceedings of the 4th ACM Conference on Recommender Systems (RecSys 2010)*, 2010, pp. 39–46
- [57] *Datasets for the evaluation of Linked Open Data Recommender Systems*. URL: <http://sisinflab.poliba.it/semanticweb/lod/recsys/datasets>. 2017. – Accessed: December 2017
- [58] DAVIES, John ; WEEKS, Richard: QuizRDF: Search technology for the Semantic Web. In: *Proceedings of the 37th Annual Hawaii International Conference on System Sciences*, 2004
- [59] *DBpedia*. URL: <http://wiki.dbpedia.org>. 2017. – Accessed: December 2017

- [60] *DBpedia - About*. URL: <http://wiki.dbpedia.org/about>. 2017. – Accessed: February 2017
- [61] *DBpedia Ontology*. URL: <http://mappings.dbpedia.org/server/ontology/classes>. – Accessed: February 2017
- [62] *DCMI metadata terms*. URL: <http://dublincore.org/documents/dcmi-terms/>. – Accessed: December 2016
- [63] DEMŠAR, Janez: Statistical comparisons of classifiers over multiple data sets. In: *Journal of Machine Learning Research* 7 (2006), no. 1, pp. 1–30
- [64] DEVITT, Ann ; VOGEL, Carl: The topology of Wordnet: Some metrics. In: *Proceedings of the 2nd International WordNet Conference (GWC)*, 2004, pp. 106–111
- [65] DI NOIA, Tommaso ; CANTADOR, Ivàn ; OSTUNI, Vito C.: Linked Open Data-enabled recommender systems: ESWC 2014 challenge on book recommendation. In: PRESUTTI, Valentina (Eds.): *Semantic Web Evaluation Challenge*. 2014, pp. 129–143
- [66] DI NOIA, Tommaso ; MIRIZZI, Roberto ; OSTUNI, Vito C. ; ROMITO, Davide: Exploiting the web of data in model-based recommender systems. In: *Proceedings of the 6th ACM Conference on Recommender Systems (RecSys 2012)*, 2012, pp. 253–256
- [67] DI NOIA, Tommaso ; MIRIZZI, Roberto ; OSTUNI, Vito C. ; ROMITO, Davide ; ZANKER, Markus: Linked Open Data to support content-based recommender systems. In: *Proceedings of the 8th International Conference on Semantic Systems*, 2012, pp. 1–8
- [68] DI NOIA, Tommaso ; OSTUNI, Vito C. ; TOMEO, Paolo ; DI SCIASCIO, Eugenio: SPRank: Semantic path-based ranking for top-n recommendations using Linked Open Data. In: *Proceedings of the 7th ACM Conference on Recommender Systems (RecSys (2013))*, 2013, pp. 86–92
- [69] DIETTERICH, Thomas G.: Ensemble methods in machine learning. In: *International Workshop on Multiple Classifier Systems*, 2000, pp. 1–15
- [70] DILLMAN, Don A. ; TORTORA, Robert D. ; BOWKER, Dennis: Principles for constructing web surveys. In: *Joint Meetings of the American Statistical Association*, 1998
- [71] DINET, Jerome ; FAVART, Monik ; PASSERAULT, Jean-Michel: Searching for information in an online public access catalogue (OPAC): the impacts of information search expertise on the use of Boolean operators. In: *Journal of Computer Assisted Learning* 20 (2004), no. 5, pp. 338–346

- [72] DING, Li ; FININ, Tim ; JOSHI, Anupam ; PAN, Rong ; COST, R. S. ; PENG, Yun ; REDDIVARI, Pavan ; DOSHI, Vishal ; SACHS, Joel: Swoogle: a search and metadata engine for the Semantic Web. In: *Proceedings of the 13th ACM International Conference on Information and Knowledge Management*, 2004, pp. 652–659
- [73] *Discovery Hub*. URL: <http://discoveryhub.co/>. 2017. – Accessed: June 2017
- [74] *Downloads 2015-10*. URL: <http://wiki.dbpedia.org/Downloads2015-10>. 2016. – Accessed: December 2016
- [75] *Downloads 2016-10*. URL: <http://wiki.dbpedia.org/Downloads2016-10>. 2016. – Accessed: December 2016
- [76] *Downloads 3.9*. URL: <http://wiki.dbpedia.org/services-resources/datasets/data-set-39/downloads-39>. 2017. – Accessed: July 2017
- [77] DÜRST, Martin ; SUIGNARD, Michel: *Internationalized resource identifiers (IRIs)*. URL: <https://tools.ietf.org/html/rfc3987>. 2004. – Accessed: December 2017
- [78] *EconStor*. URL: <https://www.econstor.eu/>. – Accessed: August 2017
- [79] *EconStor LOD*. URL: <http://www.zbw.eu/labs/en/project/econstor-lod>. – Accessed: February 2017
- [80] EL DEMERDASH, Osama ; BERGLER, Sabine ; KOSSEIM, Leila ; LANGSHAW, P. K.: Developing AMIE: an adaptive multimedia integrated environment. In: *International Workshop on Adaptive Multimedia Retrieval*, 2005, pp. 65–78
- [81] FERNÁNDEZ-TOBÍAS, Ignacio ; CANTADOR, Iván ; KAMINSKAS, Marius ; RICCI, Francesco: A generic semantic-based framework for cross-domain recommendation. In: *Proceedings of the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems*, 2011, pp. 25–32
- [82] FERNÁNDEZ-TOBÍAS, Ignacio ; TOMEIO, Paolo ; CANTADOR, Iván ; DI NOIA, Tommaso ; DI SCIASCIO, Eugenio: Accuracy and diversity in cross-domain recommendations for cold-start users with positive-only feedback. In: *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys 2016)*, 2016, pp. 119–122
- [83] (*FOAF profiles*). URL: <http://s.russwurm.org/laurelrusswurm/foaf>. 2016. – Accessed: December 2016
- [84] FOSSATI, Marco ; GIULIANO, Claudio ; TUMMARELLO, Giovanni: Semantic network-driven news recommender systems: a celebrity gossip use case. In: *Proceedings of the*

2012 International Conference on Semantic Technologies Meet Recommender Systems & Big Data, 2012, pp. 25–36

- [85] *Freebase API - data dumps*. URL: <https://developers.google.com/freebase>. 2017. – Accessed: February 2017
- [86] FREITAS, André ; OLIVEIRA, João ; O’RIAIN, Seán ; CURRY, Edward ; SILVA, João. C.: Querying Linked Data using semantic relatedness: a vocabulary independent approach. In: *International Conference on Application of Natural Language to Information Systems*, 2011, pp. 40–51
- [87] *The Gene Ontology Project - about*. URL: <http://www.geneontology.org/page/about>. 2018. – Accessed: January 2018
- [88] *Geonames*. URL: <http://www.geonames.org>. – Accessed: February 2017
- [89] *Geopolitical ontology*. URL: <http://www.geonames.org>. 2018. – Accessed: January 2018
- [90] GILES, Jim: Internet encyclopaedias go head to head. In: *Nature* 438 (2005), pp. 900–901
- [91] *GNU Free Documentation License*. URL: <https://www.gnu.org/licenses/fdl-1.3.en.html>. 2016. – Accessed: July 2017
- [92] *GNU social*. URL: <https://www.gnu.io/social>. – Accessed: February 2017
- [93] GRAINGER, Trey ; POTTER, Timothy ; SEELEY, Yonik: *Solr in action*. Manning Cherry Hill, 2014
- [94] GROUES, Valentin ; NAUDET, Yannick ; KAO, Odej: Adaptation and evaluation of a semantic similarity measure for DBpedia: a first experiment. In: *7th International Workshop on Semantic and Social Media Adaptation and Personalization (SMAP 2012)*, 2012, pp. 87–91
- [95] GUEROUSSOVA, Marina ; POLLERES, Axel ; MCILRAITH, Sheila: SPARQL with qualitative and quantitative preferences. In: *Proceedings of the 2nd International Conference on Ordering and Reasoning (OrdRing’13)* vol. 1059, 2013, pp. 2–8
- [96] HAHN, Rasmus ; BIZER, Christian ; SAHNWALDT, Christopher ; HERTA, Christian ; ROBINSON, Scott ; BÜRGLER, Michaela ; DÜWIGER, Holger ; SCHEEL, Ulrich: Faceted Wikipedia search. In: *13th International Conference on Business Information Systems (BIS 2010)* vol. 47, 2010, pp. 1–11

- [97] HAJRA, Arben ; LATIF, Atif ; TOCHTERMANN, Klaus: Retrieving and ranking scientific publications from Linked Open Data repositories. In: *Proceedings of the 14th International Conference on Knowledge Technologies and Data-driven Business*, 2014, pp. 29
- [98] HARISPE, Sébastien: *Graph-based semantic measures - technical section*. URL: http://www.semantic-measures-library.org/sml/index.php?q=doc_graph_based_advanced#rdf_compatibility. – Accessed: February 2017
- [99] HARISPE, Sébastien ; RANWEZ, Sylvie ; JANAQI, Stefan ; MONTMAIN, Jacky: Semantic measures based on RDF projections: Application to content-based recommendation systems. In: *OTM Confederated International Conferences "On the Move to Meaningful Internet Systems"*, 2013, pp. 606–615
- [100] HARISPE, Sébastien ; RANWEZ, Sylvie ; JANAQI, Stefan ; MONTMAIN, Jacky: The semantic measures library and toolkit: Fast computation of semantic similarity and relatedness using biomedical ontologies. In: *Bioinformatics* 30 (2014), no. 5, pp. 740–742
- [101] HARPER, F. M. ; KONSTAN, Joseph A.: The MovieLens datasets: History and context. In: *ACM Transactions on Interactive Intelligent Systems (TiiS)* 5 (2016), no. 4
- [102] HARRIS, Steve ; SEABORNE, Andy ; PRUD'HOMMEAUX, Eric: SPARQL 1.1 query language. In: *W3C Recommendation* 21 (2013), no. 10
- [103] HARTH, Andreas: VisiNav: a system for visual search and navigation on web data. In: *Web Semantics: Science, Services and Agents on the World Wide Web* 8 (2010), no. 4, pp. 348–354
- [104] HEATH, Tom ; BIZER, Christian: *Linked Data: Evolving the web into a global data space*. Morgan & Claypool, 2011
- [105] HEIM, Philipp ; ZIEGLER, Jürgen ; LOHMANN, Steffen: gFacet: a browser for the web of data. In: *Proceedings of the International Workshop on Interacting with Multimedia Content in the Social Semantic Web (IMC-SSW'08)* vol. 417, 2008, pp. 49–58
- [106] HEITMANN, Benjamin ; HAYES, Conor: Using Linked Data to build open, collaborative recommender systems. In: *AAAI Spring Symposium: Linked Data Meets Artificial Intelligence*, 2010, pp. 76–81
- [107] HERLOCKER, Jonathan L. ; KONSTAN, Joseph A. ; TERVEEN, Loren G. ; RIEDL, John T.: Evaluating collaborative filtering recommender systems. In: *ACM Transactions on Information Systems (TOIS)* 22 (2004), no. 1, pp. 5–53

- [108] *Hetrec2011-Lastfm-2k*. URL: <http://ir.ii.uam.es/hetrec2011/datasets/lastfm/readme.txt>. 2011. – Accessed: January 2018
- [109] HILDEBRAND, Michiel ; OSSENBRUGGEN, Jacco van ; HARDMAN, Lynda: */facet: a browser for heterogeneous Semantic Web repositories*. In: *5th International Semantic Web Conference (ISWC 2006)*, 2006, pp. 272–285
- [110] HOGAN, Aidan ; HARTH, Andreas ; UMBRICH, Jürgen ; KINSELLA, Sheila ; POLLERES, Axel ; DECKER, Stefan: Searching and browsing Linked Data with SWSE: the Semantic Web search engine. In: *Web Semantics: Science, Services and Agents on the World Wide Web* 9 (2011), no. 4, pp. 365–401
- [111] HOGAN, Aidan ; UMBRICH, Jürgen ; HARTH, Andreas ; CYGANIAK, Richard ; POLLERES, Axel ; DECKER, Stefan: An empirical survey of Linked Data conformance. In: *Web Semantics: Science, Services and Agents on the World Wide Web* 14 (2012), pp. 14–44
- [112] HU, Yajie ; WANG, Ziqi ; WU, Wei ; GUO, Jianzhong ; ZHANG, Ming: Recommendation for movies and stars using YAGO and IMDB. In: *12th International Asia-Pacific Web Conference (APWEB)*, 2010, pp. 123–129
- [113] HUTCHINS, W. J.: The concept of 'aboutness' in subject indexing. In: *Aslib Proceedings* vol. 30, 1978, pp. 172–181
- [114] HUYNH, David ; MAZZOCCHI, Stefano ; KARGER, David: Piggy bank: experience the Semantic Web inside your web browser. In: *4th International Semantic Web Conference (ISWC 2005)*, 2005, pp. 413–430
- [115] HUYNH, David F. ; KARGER, David: Parallax and companion: Set-based browsing for the data web. In: *WWW Conference ACM*, 2009
- [116] *Internet - statistics and market data about the internet*. URL: <https://www.statista.com/markets/424/internet>. – Accessed: November 2017
- [117] ISAAC, Antoine ; PHIPPS, Jon ; RUBIN, Daniel: *SKOS use cases and requirements*. URL: <https://www.w3.org/TR/skos-ucr/>. 2010. – Accessed: December 2016
- [118] *ISO/IEC/IEEE 29148: Systems and software engineering life cycle processes requirements engineering*. URL: <http://mmf.nsu.ru/sites/default/files/iso-iec-ieee-29148-2011.pdf>. 2011. – Accessed: June 2016
- [119] JACOBS, Ian ; WALSH, Norman: Architecture of the World Wide Web. In: *W3C Recommendation* 1 (2004)

- [120] JACOBSON, Kurt ; RAIMOND, Yves: *The similarity ontology*. URL: <http://grasstunes.net/ontology/musim/musim.html>. 2010. – Accessed: June 2016
- [121] JANNACH, Dietmar ; ZANKER, Markus ; FELFERNIG, Alexander ; FRIEDRICH, Gerhard: *Recommender systems: an introduction*. Cambridge University Press, 2010
- [122] JÄRVELIN, Kalervo ; KEKÄLÄINEN, Jaana: Cumulated gain-based evaluation of IR techniques. In: *ACM Transactions on Information Systems (TOIS)* 20 (2002), no. 4, pp. 422–446
- [123] *Java*. URL: <https://www.java.com>. – Accessed: December 2017
- [124] *Java CC - the Java parser generator*. URL: <https://javacc.org>. – Accessed: December 2017
- [125] *Java servlet technology overview*. URL: <http://www.oracle.com/technetwork/java/javaee/servlet/index.html>. – Accessed: December 2017
- [126] *JQuery.ajax()*. URL: <http://api.jquery.com/jquery.ajax>. – Accessed: November 2017
- [127] KABUTOYA, Yutaka ; SUMI, Robert ; IWATA, Tomoharu ; UCHIYAMA, Toshio ; UCHIYAMA, Tadasu: A topic model for recommending movies via Linked Open Data. In: *IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, 2012, pp. 625–630
- [128] KAMINSKAS, Marius ; FERNÁNDEZ-TOBÍAS, Ignacio ; RICCI, Francesco ; CANTADOR, Iván: Knowledge-based music retrieval for places of interest. In: *Proceedings of the 2nd International ACM Workshop on Music Information Retrieval with User-centered and Multimodal Strategies*, 2012, pp. 19–24
- [129] KHROUF, Houda ; TRONCY, Raphaé: Hybrid event recommendation using Linked Data and user diversity. In: *Proceedings of the 7th ACM Conference on Recommender Systems (RecSys 2013)*, 2013, pp. 185–192
- [130] KIEFER, Christoph ; BERNSTEIN, Abraham ; STOCKER, Markus: The fundamentals of iSPARQL: a virtual triple approach for similarity-based semantic web tasks. In: *The Semantic Web*. Springer, 2007, pp. 295–309
- [131] KLAHOLD, André: *Empfehlungssysteme*. Vieweg+Teubner, 2009

- [132] KNIJNENBURG, Bart: *Recommender systems and the user-centric evaluation of their interfaces*. URL: <https://www.usabart.nl/portfolio/framework.pdf>. – Accessed: July 2017
- [133] KNIJNENBURG, Bart P. ; WILLEMSSEN, Martijn C.: Evaluating recommender systems with user experiments. In: LOPS, Pasquale (Eds.): *Recommender Systems Handbook*. Springer, 2015, pp. 309–352
- [134] KNIJNENBURG, Bart P. ; WILLEMSSEN, Martijn C. ; GANTNER, Zeno ; SONCU, Hakan ; NEWELL, Chris: Explaining the user experience of recommender systems. In: *User Modeling and User-Adapted Interaction 22* (2012), no. 4/5, pp. 441–504
- [135] KOBILAROV, Georgi ; DICKINSON, Ian: Humboldt: Exploring Linked Data. In: *Proceedings of the WWW2008 Workshop on Linked Data on the Web (LDOW 2008)*, 2008
- [136] KOIVUNEN, Marja-Riitta ; MILLER, Eric: W3C semantic web activity. In: *Semantic Web Kick-off Seminar in Finland*, 2001, pp. 27–44
- [137] KONSTAN, Joseph A. ; RIEDL, John: Recommender systems: from algorithms to user experience. In: *User Modeling and User-Adapted Interaction 22* (2012), no. 1, pp. 101–123
- [138] KONTOKOSTAS, Dimitris ; WESTPHAL, Patrick ; AUER, Sören ; HELLMANN, Sebastian ; LEHMANN, Jens ; CORNELISSEN, Roland ; ZAVERI, Amrapali: Test-driven evaluation of Linked Data quality. In: *Proceedings of the 23rd International Conference on World Wide Web*, 2014, pp. 747–758
- [139] KOREN, Yehuda ; BELL, Robert M. ; VOLINSKY, Chris: *The BellKor solution to the Netflix prize*. URL: <https://pdfs.semanticscholar.org/f4eb/b542c752a0dc423f94fd121e2edb8f6275ba.pdf>. 2009. – Accessed: March 2017
- [140] KOUTRIKA, Georgia ; BERCOVITZ, Benjamin ; GARCIA-MOLINA, Hector: FlexRecs: Expressing and combining flexible recommendations. In: *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, 2009, pp. 745–758
- [141] KULES, Bill ; CAPRA, Robert ; BANTA, Matthew ; SIERRA, Tito: What do exploratory searchers look at in a faceted search interface? In: *Proceedings of the 9th ACM/IEEE-CS Joint Conference on Digital Libraries*, 2009, pp. 313–322
- [142] LEE, Myungjin ; KIM, Wooju: Semantic association search and rank method based on spreading activation for the Semantic Web. In: *IEEE International Conference on Industrial Engineering and Engineering Management*, 2009, pp. 1523–1527

- [143] LEHMANN, Jens ; BÜHMANN, Lorenz: AutoSPARQL: Let users query your knowledge base. In: *The Semantic Web: Research and Applications* (2011), pp. 63–79
- [144] LEHMANN, Jens ; ISELE, Robert ; JAKOB, Max ; JENTZSCH, Anja ; KONTOKOSTAS, Dimitris ; MENDES, Pablo N. ; HELLMANN, Sebastian ; MORSEY, Mohamed ; VAN KLEEF, Patrick ; AUER, Sören ; BITZER, Christian: DBpedia: a large-scale, multilingual knowledge base extracted from Wikipedia. In: *Semantic Web 6* (2015), no. 2, pp. 167–195
- [145] *Lexvo.org - about*. URL: <http://www.lexvo.org>. 2017. – Accessed: February 2017
- [146] LIN, Dekang: An information-theoretic definition of similarity. In: *Proceedings of the 15th International Conference on Machine Learning (ICML'98)* vol. 98, 1998, pp. 296–304
- [147] LINDEN, Greg ; SMITH, Brent ; YORK, Jeremy: Amazon.com recommendations: Item-to-item collaborative filtering. In: *IEEE Internet Computing* 7 (2003), no. 1, pp. 76–80
- [148] *The Linking Open Data cloud diagram*. URL: <https://lod-cloud.net>. 2017. – Accessed: March 2017
- [149] *LODStats*. URL: <http://stats.lod2.eu>. – Accessed: February 2017,
- [150] *LODStats - vocabularies*. URL: <https://stats.lod2.eu/vocabularies>. – Accessed: February 2017
- [151] *LODStats - a statement-stream-based approach for gathering comprehensive statistics about RDF datasets*. URL: <https://www.netflix.com>. – Accessed: February 2017
- [152] LOPS, Pasquale ; DE GEMMIS, Marco ; SEMERARO, Giovanni: Content-based recommender systems: State of the art and trends
- [153] LOWE, Harry J. ; BARNETT, G. O.: Understanding and using the medical subject headings (MeSH) vocabulary to perform literature searches. In: *JAMA: The Journal of the American Medical Association* 271 (1994), no. 14, pp. 1103–1108
- [154] MACGREGOR, George ; MCCULLOCH, Emma: Collaborative tagging as a knowledge organisation and resource discovery tool. In: *Library Review* 55 (2006), no. 5, pp. 291–300
- [155] MADER, Christian ; HASLHOFER, Bernd ; ISAAC, Antoine: Finding quality issues in SKOS vocabularies. In: *International Conference on Theory and Practice of Digital Libraries*, 2012, pp. 222–233

- [156] MAEDCHE, Alexander ; STAAB, Steffen: Measuring similarity between ontologies. In: *International Conference on Knowledge Engineering and Knowledge Management*, 2002, pp. 251–263
- [157] MANNENS, Erik ; COPPENS, Sam ; DE PESSEMIER, Toon ; DACQUIN, Hendrik ; VAN DEURSEN, Davy ; DE SUTTER, Robbie ; WALLE, Rik Van d.: Automatic news recommendations via aggregated profiling. In: *Multimedia Tools and Applications* 63 (2013), no. 2, pp. 407–425
- [158] MANNING, Christopher D. ; RAGHAVAN, Prabhakar ; SCHÜTZE, Hinrich: *Introduction to information retrieval*. Cambridge university press Cambridge, 2008
- [159] *Mapping artists of Lastfm to DBpedia*. URL: <https://github.com/sisinflab/LODrecsys-datasets/tree/master/LastFM>. 2017. – Accessed: January 2018
- [160] *Mapping artists of MovieLens1M to DBpedia*. URL: <https://github.com/sisinflab/LODrecsys-datasets/tree/master/MovieLens1M>. 2017. – Accessed: January 2018
- [161] *Mapping books of LibraryThing to DBpedia*. URL: <https://github.com/sisinflab/LODrecsys-datasets/tree/master/LibraryThing>. 2017. – Accessed: January 2018
- [162] MARIE, Nicolas ; GANDON, Fabien ; RIBIÈRE, Myriam ; RODIO, Florentin: Discovery hub: on-the-fly Linked Data exploratory search. In: *Proceedings of the 9th International Conference on Semantic Systems*, 2013, pp. 17–24
- [163] MAYR, Philipp ; PETRAS, Vivien: Cross-concordances: terminology mapping and its effectiveness for information retrieval. In: *ArXiv Preprint arXiv:0806.3765* (2008)
- [164] MCNEE, Sean M. ; RIEDL, John ; KONSTAN, Joseph A.: Being accurate is not enough: how accuracy metrics have hurt recommender systems. In: *Extended Abstracts on Human Factors in Computing Systems (CHI'06)*, 2006, pp. 1097–1101
- [165] MEYMANDPOUR, Rouzbeh ; DAVIS, Joseph G.: Recommendations using Linked Data. In: *Proceedings of the 5th Ph.D. Workshop on Information and Knowledge*, 2012, pp. 75–82
- [166] MILES, Alistair ; BECHHOFFER, Sean: SKOS Simple Knowledge Organization System reference. In: *W3C Recommendation* (2009)
- [167] MISTRY, Meeta ; PAVLIDIS, Paul: Gene Ontology term overlap as a measure of gene functional similarity. In: *BMC Bioinformatics* 9 (2008), no. 1

- [168] MOBASHER, Bamshad ; JIN, Xin ; ZHOU, Yanzan: Semantically enhanced collaborative filtering on the web. In: *Web Mining: From Web to Semantic Web*. Springer, 2004, pp. 57–76
- [169] MORSEY, Mohamed ; LEHMANN, Jens ; AUER, Sören ; STADLER, Claus ; HELLMANN, Sebastian: DBpedia and the live extraction of structured data from Wikipedia. In: *Program* 46 (2012), no. 2, pp. 157–181
- [170] *MovieLens 1M dataset*. URL: <https://grouplens.org/datasets/movielens/1m>. 2018. – Accessed: January 2018
- [171] MUSETTI, Alberto ; NUZZOLESE, Andrea G. ; DRAICCHIO, Francesco ; PRESUTTI, Valentina ; BLOMQVIST, Eva ; GANGEMI, Aldo ; CIANCARINI, Paolo: Aemoo: Exploratory search based on knowledge patterns over the Semantic Web. In: *Semantic Web Challenge at the 10th International Semantic Web Conference (ISWC 2011)*, 2011
- [172] *Netflix*. URL: <https://www.netflix.com>. 2018. – Accessed: March 2018
- [173] NEUBERT, Joachim: Bringing the "Thesaurus for Economics" on to the web of Linked Data. In: *Proceedings of the WWW2009 Workshop on Linked Data on the Web (LDOW 2009)* (2009)
- [174] NEUMANN, Thomas ; WEIKUM, Gerhard: The RDF-3X engine for scalable management of RDF data. In: *The International Journal on Very Large Data Bases (VLDB)* 19 (2010), no. 1, pp. 91–113
- [175] *Open Database License (ODbL) v1.0*. URL: <https://opendatacommons.org/licenses/odbl/1-0/>. – Accessed: February 2017
- [176] *Open Definition 2.1*. (2014). – Accessed: March 2017
- [177] OREN, Eyal ; DELBRU, Renaud ; CATASTA, Michele ; CYGANIAK, Richard ; STENZHORN, Holger ; TUMMARELLO, Giovanni: Sindice. com: a document-oriented lookup index for open Linked Data. In: *International Journal of Metadata, Semantics and Ontologies* 3 (2008), no. 1, pp. 37–52
- [178] OREN, Eyal ; DELBRU, Renaud ; DECKER, Stefan: Extending faceted navigation for RDF data. In: *5th International Semantic Web Conference (ISWC 2006)*, 2006, pp. 559–572
- [179] OSTUNI, Vito C. ; DI NOIA, Tommaso ; DI SCIASCIO, Eugenio ; MIRIZZI, Roberto: Top-n recommendations from implicit feedback leveraging Linked Open Data. In: *Proceedings of the 7th ACM Conference on Recommender Systems (RecSys 2013)*, 2013, pp. 85–92

- [180] OZDIKIS, Ozer ; ORHAN, Fatih ; DANISMAZ, Furkan: Ontology-based recommendation for points of interest retrieved from multiple data sources. In: *Proceedings of the International Workshop on Semantic Web Information Management*, 2011
- [181] PARK, Jung-ran: Metadata quality in digital repositories: A survey of the current state of the art. In: *Cataloging & Classification Quarterly* 47 (2009), no. 3/4, pp. 213–228
- [182] PASSANT, Alexandre: Dbrec: music recommendations using DBpedia. In: *9th International Semantic Web Conference (ISWC 2010)*, 2010, pp. 209–224
- [183] PASSANT, Alexandre ; RAIMOND, Yves: Combining social music and Semantic Web for music-related recommender systems. In: *Social Data on the Web Workshop*, 2008
- [184] PEFFERS, Ken ; TUUNANEN, Tuure ; ROTHENBERGER, Marcus A. ; CHATTERJEE, Samir: A design science research methodology for information systems research. In: *Journal of Management Information Systems* 24 (2007), no. 3, pp. 45–77
- [185] PEREZ, Jorge ; ARENAS, Marcelo ; GUTIERREZ, Claudio: Semantics and complexity of SPARQL. In: *ACM Transactions on Database Systems (TODS)* 34 (2009), no. 3, pp. 16
- [186] PESKA, Ladislav ; VOJTÁS, Peter: Enhancing recommender system with Linked Open Data. In: *International Conference on Flexible Query Answering Systems*, 2013, pp. 483–494
- [187] PESQUITA, Catia ; FARIA, Daniel ; FALCAO, André O. ; LORD, Phillip ; COUTO, Francisco M.: Semantic similarity in biomedical ontologies. In: *PLoS Computational Biology* 5 (2009), no. 7
- [188] POHL, Adrian ; DANOWSKI, Patrick: Linked Open Data in der Bibliothekswelt: Grundlagen und Überblick. In: DANOWSKI, Patrick (Eds.): *(Open) Linked Data in Bibliotheken*
- [189] POLICARPIO, Sean ; BRUNK, Soren ; TUMMARELLO, Giovanni: Implementation of a SPARQL integrated recommendation engine for Linked Data with hybrid capabilities. In: *Artificial Intelligence Meets the Web of Data Workshop*, 2012
- [190] POLLERES, Axel ; HOGAN, Aidan ; DELBRU, Renaud ; UMBRICH, Jürgen: RDFS and OWL reasoning for Linked Data. In: *Reasoning Web: Semantic Technologies for Intelligent Data Access*. Springer, 2013, pp. 91–149
- [191] PRUD'HOMMEAUX, Eric ; BUIL-ARANDA, Carlos ; SEABORNE, Andy ; POLLERES, Axel ; FEIGENBAUM, Lee ; WILLIAMS, Gregory T.: SPARQL 1.1 federated query. In: *W3C Recommendation* (2013)

- [192] PRUD'HOMMEAUX, Eric ; SEABORNE, Andy: SPARQL query language for RDF. In: *W3C Recommendation 15* (2008)
- [193] PU, Pearl ; CHEN, Li ; HU, Rong: A user-centric evaluation framework for recommender systems. In: *Proceedings of the 5th ACM Conference on Recommender Systems (RecSys 2011)*, 2011, pp. 157–164
- [194] PU, Pearl ; CHEN, Li ; HU, Rong: Evaluating recommender systems from the user's perspective: survey of the state of the art. In: *User Modeling and User-Adapted Interaction 22* (2012), no. 4, pp. 317–355
- [195] *Public Domain dedication - CC0 1.0 Universal (CC0 1.0)*. URL: <https://creativecommons.org/publicdomain/zero/1.0>. – Accessed: August 2017
- [196] *Quentin Tarantino*. URL: https://en.wikipedia.org/wiki/Quentin_Tarantino. 2017. – Accessed: April 2017
- [197] (*Query limits for DBpedia*). URL: <https://groups.google.com/forum/#!topic/thosch/ApzhPjQZlvg>. – Accessed: July 2015
- [198] QUILITZ, Bastian: *DARQ: federated queries with SPARQL*. URL: <http://darq.sourceforge.net/>. 2006. – Accessed: March 2017
- [199] RADA, Roy ; MILI, Hafdeh ; BICKNELL, Ellen ; BLETTNER, Maria: Development and application of a metric on semantic nets. In: *IEEE Transactions on Systems, Man, and Cybernetics 19* (1989), no. 1, pp. 17–30
- [200] RAKHMAWATI, Nur A. ; UMBRICH, Jürgen ; KARNSTEDT, Marcel ; HASNAIN, Ali ; HAUSENBLAS, Michael: Querying over federated SPARQL endpoints: a state of the art survey. In: *ArXiv Preprint ArXiv:1306.1723* (2013)
- [201] RESNICK, Paul ; IACOVOU, Neophytos ; SUCHAK, Mitesh ; BERGSTROM, Peter ; RIEDL, John: GroupLens: an open architecture for collaborative filtering of netnews. In: *Proceedings of the 1994 ACM Conference on Computer-supported Cooperative Work*, 1994, pp. 175–186
- [202] RESNIK, Philip: Using information content to evaluate semantic similarity in a taxonomy. In: *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)* vol. 1, 1995, pp. 448–453
- [203] *Revyu.com - review anything*. URL: <http://revyu.com>. – Accessed: February 2017
- [204] RICCI, Francesco ; ROKACH, Lior ; SHAPIRA, Bracha: Recommender systems: Introduction and challenges

- [205] RIEMER, Kai ; BRÜGGEMANN, Fabian: Personalisierung der Internetsuche. In: *Wirtschaftsinformatik* 49 (2007), no. 2, pp. 116–126
- [206] RISTOSKI, Petar ; MENCÍA, Eneldo L. ; PAULHEIM, Heiko: A hybrid multi-strategy recommender system using Linked Open Data. In: PRESUTTI, Valentina (Eds.): *Semantic Web Evaluation Challenge*, 2014, pp. 150–156
- [207] ROSATI, Jessica ; DI NOIA, Tommaso ; LUKASIEWICZ, Thomas ; DE LEONE, Renato ; MAURINO, Andrea: Preference queries with ceteris paribus semantics for Linked uot-saata. In: *OTM Confederated International Conferences "On the Move to Meaningful Internet Systems"*, 2015, pp. 423–442
- [208] RUOTSALO, Tuukka ; HAAV, Krister ; STOYANOV, Antony ; ROCHE, Sylvain ; FANI, Elena ; DELIAI, Romina ; MÄKELÄ, Eetu ; KAUPPINEN, Tomi ; HYVÖNEN, Eero: SMARTMUSEUM: A mobile recommender system for the web of data. In: *Web Semantics: Science, Services and Agents on the World Wide Web* 20 (2013), pp. 50–67
- [209] RUPP, Chris: *Requirements-Engineering und-Management: Aus der Praxis von klassisch bis agil*. Carl Hanser Verlag, 2014
- [210] RUTHVEN, Ian ; LALMAS, Mounia ; VAN RIJSBERGEN, Keith: Incorporating user search behavior into relevance feedback. In: *Journal of the Association for Information Science and Technology* 54 (2003), no. 6, pp. 529–549
- [211] *Sample consent form for a web-based study*. URL: <https://www.socialpsychology.org/consentform.htm>. – Accessed: November 2017
- [212] SARWAR, Badrul ; KARYPIS, George ; KONSTAN, Joseph ; RIEDL, John: Item-based collaborative filtering recommendation algorithms. In: *Proceedings of the 10th International Conference on World Wide Web*, 2001, pp. 285–295
- [213] SAWERS, Paul: *Remember Netflix's 1m dollar algorithm contest? Well, here's why it didn't use the winning entry*. URL: <https://thenextweb.com>. 2012. – Accessed: July 2017
- [214] SCHCLAR, Alon ; TSIKINOVSKY, Alexander ; ROKACH, Lior ; MEISELS, Amnon ; ANTWARG, Liat: Ensemble methods for improving the performance of neighborhood-based collaborative filtering. In: *Proceedings of the 3rd ACM Conference on Recommender Systems (RecSys 2009)*, 2009, pp. 261–264
- [215] SCHMACHTENBERG, Max ; BIZER, Christian ; PAULHEIM, Heiko: Adoption of the Linked Data best practices in different topical domains. In: *13th International Semantic Web Conference (ISWC 2014)*, 2014, pp. 245–260

- [216] SCHREIBER, Guus ; RAIMOND, Yves: RDF 1.1 Primer. In: *W3C Working Group Note* (2014)
- [217] SCHULTZ, Andreas ; MATTEINI, Andrea ; ISELE, Robert ; BIZER, Christian ; BECKER, Christian: LDIF: Linked Data integration framework. In: *Proceedings of the 2nd International Conference on Consuming Linked Data*, 2011, pp. 125–130
- [218] SEABORNE, Andy: SPARQL 1.1 property paths. In: *W3C Working Draft* (2010)
- [219] SEAN, Lim Y. ; SADANANDAN, Arun A. ; LUKOSELAND, Dickson ; TOCHTERMANN, Klaus: Scientific Publication Retrieval in Linked Data
- [220] *Search for datasets*. URL: <https://datahub.io/dataset?q=format-skos>. 2016. – Accessed: November 2016
- [221] SECO, Nuno ; VEALE, Toni ; HAYES, Jer: An intrinsic information content metric for semantic similarity in WordNet. In: *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI'04)*, 2004, pp. 1089–1090
- [222] SEGARAN, Toby ; EVANS, Colin ; TAYLOR, Jamie: *Programming the Semantic Web: Build flexible applications with graph data*. O'Reilly, 2009
- [223] SHANGGUAN, Zhenning: *Installing and managing virtuoso SPARQL endpoint*. URL: https://logd.tw.rpi.edu/tutorial/installing_using_virtuoso_sparql_endpoint. – Accessed: December 2016
- [224] SHANI, Guy ; GUNAWARDANA, Asela: Evaluating recommendation systems.
- [225] SHARDANAND, Upendra ; MAES, Pattie: Social information filtering: algorithms for automating “word of mouth”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 1995, pp. 210–217
- [226] SHEKARPOUR, Saeedeh ; HÖFFNER, Konrad ; LEHMANN, Jens ; AUER, Soren: Keyword query expansion on Linked Data using linguistic and semantic features. In: *IEEE 7th International Conference on Semantic Computing (ICSC)*, 2013, pp. 191–197
- [227] SIBERSKI, Wolf ; PAN, Jeff Z. ; THADEN, Uwe: Querying the Semantic Web with preferences. In: *5th International Semantic Web Conference (ISWC 2006)* vol. 4273, 2006, pp. 612–624
- [228] SIMON, Agnès ; WENZ, Romain ; MICHEL, Vincent ; DI MASCIO, Adrien: Publishing bibliographic records on the web of data: Opportunities for the BnF (French National Library). In: *10th Extended Semantic Web Conference (ESWC 2013)*, 2013, pp. 563–577

- [229] *SKOS datasets*. URL: <https://www.w3.org/2001/sw/wiki/SKOS/Datasets>. 2015. – Accessed: December 2016
- [230] (*SPARQL 1.1 grammar*). URL: https://github.com/apache/jena/blob/master/jena-arq/Grammar/sparql_11.jj. 2016. – Accessed: December 2017
- [231] *SPARQL implementations*. URL: <https://www.w3.org/wiki/SparqlImplementations>. 2016. – Accessed: February 2017
- [232] *Standard-Thesaurus Wirtschaft*. URL: <http://zbw.eu/stw/version/latest/about.de.html>. 2017. – Accessed: August 2017
- [233] STANKOVIC, Milan ; BREITFUSS, Werner ; LAUBLET, Philippe: Discovering relevant topics using DBpedia: Providing non-obvious recommendations. In: *IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, 2011, pp. 219–222
- [234] STANKOVIC, Milan ; JOVANOVIĆ, Jelena ; LAUBLET, Philippe: Linked Data metrics for flexible expert search on the open web. In: *The Semantic Web: Research and Applications* (2011), pp. 108–123
- [235] SUMMERS, Ed ; ISAAC, Antoine ; REDDING, Clay ; KRECH, Dan: LCSH, SKOS and Linked Data. In: *Proceedings of the International Conference on Dublin Core and Metadata Applications*, 2008, pp. 25
- [236] SWEARINGEN, Kirsten ; SINHA, Rashmi: Beyond algorithms: an HCI perspective on recommender systems. In: *ACM SIGIR Workshop on Recommender Systems* vol. 13, 2001, pp. 1–11
- [237] TAN, Pang-Ning ; STEINBACH, Michael ; KUMAR, Vipin: *Introduction to data mining*. Pearson Education India, 2006
- [238] TAVAKOL, Mohsen ; DENNICK, Reg: Making sense of Cronbach’s alpha. In: *International Journal of Medical Education* 2 (2011), pp. 53–55
- [239] TUNKELANG, Daniel: Faceted search. (2009)
- [240] UMBRICH, Jürgen ; HOSE, Katja ; KARNSTEDT, Marcel ; HARTH, Andreas ; POLLERES, Axel: Comparing data summaries for processing live queries over Linked Data. In: *World Wide Web* 14 (2011), no. 5/6, pp. 495–544
- [241] UNGER, Christina ; BÜHMANN, Lorenz ; LEHMANN, Jens ; NGONGA NGOMO, Axel-Cyrille ; GERBER, Daniel ; CIMIANO, Philipp: Template-based question answering over

- RDF data. In: *Proceedings of the 21st International Conference on World Wide Web*, 2012, pp. 639–648
- [242] *Use case AGRIS*. URL: https://www.w3.org/2005/Incubator/lld/wiki/Use_Case_AGRIS. 2010. – Accessed: February 2017
- [243] VARGA, Bernadette ; GROZA, Adrian: Integrating DBpedia and SentiWordNet for a tourism recommender system. In: *IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*, 2011, pp. 133–136
- [244] VARNHAGEN, Connie K. ; GUSHTA, Matthew ; DANIELS, Jason ; PETERS, Tara C. ; PARMAR, Neil ; LAW, Danielle ; HIRSCH, Rachel ; SADLER TAKACH, Bonnie ; JOHNSON, Tom: How informed is online informed consent? In: *Ethics & Behavior* 15 (2005), no. 1, pp. 37–48
- [245] *Virtuoso SPARQL query editor*. URL: <http://dbpedia.org/sparql>. 2018. – Accessed: March 2018
- [246] WAITELONIS, Jörg ; SACK, Harald: Towards exploratory video search using Linked Data. In: *11th IEEE International Symposium on Multimedia (ISM'09)*, 2009, pp. 540–545
- [247] WARDHANA, Alexander T. A. ; NUGROHO, H. T.: Combining FOAF and music ontology for music concerts recommendation on Facebook application. In: *Conference on New Media Studies (CoNMedia)*, 2013, pp. 1–5
- [248] *Weka 3 - data mining software in Java*. URL: <https://www.cs.waikato.ac.nz/ml/weka>. 2017. – Accessed: June 2017
- [249] WENIGE, Lisa ; RUHLAND, Johannes: Scalable property aggregation for Linked Data recommender systems. In: *3rd International Conference on Future Internet of Things and Cloud (FiCloud)*, 2015, pp. 451–456
- [250] WENIGE, Lisa ; RUHLAND, Johannes: Flexible on-the-fly recommendations from Linked Open Data repositories. In: *19th International Conference on Business Information Systems (BIS 2016)*, 2016, pp. 43–54
- [251] WENIGE, Lisa ; RUHLAND, Johannes: Retrieval by recommendation: using LOD technologies to improve digital library search. In: *International Journal on Digital Libraries* (2017), pp. 1–17
- [252] *[Wikipedia] Help - category*. URL: <https://en.wikipedia.org/wiki/Help:Category>. 2018. – Accessed: July 2017

- [253] [Wikipedia] *Online access*. URL: <http://wiki.dbpedia.org/OnlineAccess>. 2017. – Accessed: July 2017
- [254] WILSON, Max ; RUSSELL, Alistair ; SMITH, Daniel A.: mSpace: Improving information access to multimedia domains with multimodal exploratory search. In: *Communications of the ACM* 49 (2006), no. 4, pp. 47–49
- [255] *Working with SPARQL endpoint constraints via LIMIT & OFFSET*. URL: <http://vos.openlinksw.com/owiki/wiki/VOS/VirtTipsAndTricksHowToHandleBandwidthLimitExceed>. – Accessed: December 2016
- [256] WU, Mingrui: Collaborative filtering via ensembles of matrix factorizations. In: *Proceedings of KDD Cup and Workshop, 2007*
- [257] WU, Zhibiao ; PALMER, Martha: Verbs semantics and lexical selection. In: *Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics*, 1994, pp. 133–138
- [258] *Yago - a high-quality knowledge base*. URL: <https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago>. 2017. – Accessed: February 2017
- [259] ZARRINKALAM, Fattane ; KAHANI, Mohsen: A multi-criteria hybrid citation recommendation system based on Linked Data. In: *2nd International eConference on Computer and Knowledge Engineering (ICCKE)*, 2012, pp. 283–288
- [260] ZHANG, Mi ; HURLEY, Neil: Avoiding monotony: Improving the diversity of recommendation lists. In: *Proceedings of the 2nd ACM Conference on Recommender Systems (RecSys 2008)*, 2008, pp. 123–130
- [261] ZIEGLER, Cai-Nicolas: Semantic web recommender systems. In: *International Conference on Extending Database Technology*, 2004, pp. 78–89
- [262] ZIEGLER, Cai-Nicolas ; MCNEE, Sean M. ; KONSTAN, Joseph A. ; LAUSEN, Georg: Improving recommendation lists through topic diversification. In: *Proceedings of the 14th International Conference on World Wide Web*, 2005, pp. 22–32
- [263] ZOU, Lei ; HUANG, Ruizhe ; WANG, Haixun ; YU, Jeffrey X. ; HE, Wenqiang ; ZHAO, Dongyan: Natural language question answering over RDF: a graph data driven approach. In: *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, 2014, pp. 313–324