Flexible On-the-Fly Recommendations from Linked Open Data Repositories

Lisa Wenige(✉) and Johannes Ruhland

Chair of Business Information Systems,
Friedrich-Schiller-University, Jena, Germany
{lisa.wenige,johannes.ruhland}@uni-jena.de

Abstract. Recommender systems help consumers to find products online. But because many content-based systems work with insufficient data, recent research has focused on enhancing item feature information with data from the Linked Open Data cloud. Linked Data recommender systems are usually bound to a predefined set of item features and offer limited opportunities to tune the recommendation model to individual needs. The paper addresses this research gap by introducing the prototype SKOS Recommender (SKOSRec), which produces scalable on-the-fly recommendations through SPARQL-like queries from Linked Data repositories. The SKOSRec query language enables users to obtain constraint-based, aggregation-based and cross-domain recommendations, such that results can be adapted to specific business or customer requirements.

Keywords: Linked Data · Recommender systems · Query-based recommender systems

1 Introduction

Recommender systems (RS) are on the forefront of decision support systems within e-commerce applications [1,2]. Among the most known examples of recommender systems are the ones used by well-established e-commerce retailers such as Amazon[1] or Netflix[2]. Here, users receive personalized suggestions for items of the product catalog. But both content-based and collaborative filtering systems suffer from certain shortcomings, such as rating sparsity or limited content analysis [1]. To address the problem of data sparsity, the Linked Open Data (LOD) movement gave rise to the type of Linked Data recommender systems (LDRS). These systems tackle drawbacks of traditional approaches by enriching existing recommender systems with information from public data sources. But even though current LDRS show promising results, they do not yet take full advantage of the potential that LOD offers. The paper addresses this research gap through:

[1] http://www.amazon.de.
[2] http://www.netflix.com.

W. Abramowicz et al. (Eds.): BIS 2016, LNBIP 255, pp. 43–54, 2016.
DOI: 10.1007/978-3-319-39426-8_4

- Identification of central problems of different types of recommender systems (Sect. 2).
- Overview of current challenges for RS in e-commerce applications and how they can be addressed with the help of Linked Data technologies (Sect. 3).
- Technical description of the SKOSRec prototype, that implements these ideas (Sect. 4).
- Evaluation of the systems' key components (Sect. 5).
- Discussion of the main findings and of the practical implications of the approach (Sect. 6).

2 Related Work

2.1 Recommender Systems

Upon the presentation of the first systems in the 1990s, the area of recommender systems has been an established research field. The most common recommendation algorithms apply some form of collaborative filtering technique, where users are referred to preferred items of like-minded peers [3–5]. In contrast to that, content-based systems derive recommendations from feature information for items in the user profile [1,6,7,9,10]. Thus, similar items are detected. Content-based and collaborative filtering systems suffer from certain shortcomings. Even though both paradigms can be combined in hybrid systems [7], many of the problems still remain.

One of the main issues on the operational side is the data sparsity problem, where user preferences are rare. It mostly occurrs, when new users or items are added to the system (cold start problem), but it can also arise when the amount of feedback information is simply not enough to derive meaningful recommendations. Especially in content-based systems, users can receive unfitting recommendations due to incomplete or ambiguous item feature information (limited content analysis) [1].

2.2 Linked Data Recommender Systems

Recently, researchers have started to utilize Linked Data information sources to address the problem of insufficient item feature information. The LOD cloud comprises data on almost any kind of subject and offers general purpose information (e.g. DBpedia[3]) as well as data from special domains [8]. LOD resources are usually identified through URIs. Thus, the LOD cloud provides less ambiguous information than text-based item descriptions [9]. Experiments on historic data showed that LDRS are at least competitive with classical systems and sometimes even outperform them in terms of accuracy [9–12,21].

But even though LDRS achieve considerable results, they do not yet take full advantage of the LOD cloud. Current approaches require a considerable amount

[3] http://wiki.dbpedia.org/.

of pre-processing, such as the selection and extraction of item features. Once a set of item features has been selected, the recommendation model is 'hard wired' into the system and can not be adapted to changing user or business demands.

2.3 Query-Based Recommender Systems

Due to the fact that most LDRS and non-LDRS are not capable of customizations, there have been efforts to enable systems to produce query-based recommendations. In the field of non-LDRS this is achieved through enhancing relational databases with recommendation functionalities [13]. For instance, the REQUEST system integrates the personalization process with OLAP-like queries, such that the selection of items/users can be based on certain conditions and aggregations. Thus, recommendation models are adaptable to different requirements at runtime [14,15]. But as information on user preferences is usually sparse, this information becomes even sparser when only certain items or users are selected. This often leads to unreliable recommendation results [15].

The issue of data sparsity could be addressed through content-based approaches that enhance item feature information with LOD resources. To date, there are only a few systems that consider user preferences in conjunction with Linked Data technologies, such as SPARQL queries [16–19]. With these systems, expressive user preferences can be formulated. But we argue that the potential of LOD is not yet fully exploited for recommendation tasks. Query-based LDRS either follow a fixed workflow of similarity detection and SPARQL graph pattern matchings [16,17] or face long execution times when processing a large number of triple statements [18].

Therefore, the main goal of the paper is the development of a system that flexibly integrates user preferences with SPARQL elements at reasonable computational cost and hat provides novel and meaningful recommendations in update-intensive environments, where local databases do not provide sufficient data.

3 LOD for Flexible Recommendations in e-Commerce

The following aspects give an overview of current challenges of RS applications in the e-commerce sector and describe how they can be addressed with the help of Linked Open Data:

– **Comprehensiveness:** As stated in the previous sections, e-commerce RS have to deal with the issues of data sparsity and low profile overlap among users. Especially small sites do not have a customer base that is big enough to provide enough ratings [2]. That is why the integration of Linked Data sources into recommender systems could help to overcome existing limitations on the data side. The LOD cloud comprises billions of triple statements ranging from general purpose data to information sources from domains, such as media or geography. These datasets can be of value in multimedia retailing or online travel agencies [8].

- **Adaptability:** RS of e-commerce sites usually do not offer functionalities where customers can restrict recommendation results to specific criteria. But to achieve deeply personalized results, it would be desirable to apply pre- or postfiltering on the product catalog [2,13]. For instance, think of a customer of a media streaming site who, in spite of his/her purchasing history, wants to provide the information that he/she is strongly interested in European movies. Above that, in areas like tourism, user preferences depend on many factors, such as context, travel companions or travel destination preferences [20]. In addition to that, not only consumers could profit from customization functionalities, but also marketing professionals and administrators [2,15]. For instance, marketing campaigns could be fit to special holiday occassions of the year to promote long-tail items. These aspects require data-rich applications, that can be accessed with expressive queries.
- **On-the-fly recommendations:** Current RS usually rely on pre-computed recommendation results. But the aspect of adapatability is strongly tied to the aspect of just-in-time recommendations. As customer and business requirements can not be foreseen, recommendation models should be configurable at runtime, such that a user can select the right data when it is actually needed [21]. To enable flexible recommendations results from Linked Open Data repositories, efficient strategies for processing large numbers of triple statements have to be identified.

4 The SKOS Recommender

In this section we present SKOSRec, a system prototype that addresses the previously identified challenges of RS in e-commerce applications.

4.1 Scalable On-the-Fly Recommendations

Most LDRS identify similar items through their features. But considering the large amount of information, using all known features of a resource leads to poor scalability and long processing times. Thus, in the context of LDRS it was proposed to select certain item features (properties) for the recommendation model [9,10,22–24]. But due to the large amount of information in the LOD cloud, the selection process can be error-prone and time consuming. Thus we propose to perform similarity computation on URI annotations that are part of commonly used vocabularies, such as the Simple Knowledge Organization System (SKOS). SKOS vocabularies have become a de-facto standard for subject annotations, since a majority of Linked Data sets are annotated with SKOS concepts. We implemented a system, called SKOS Recommender that uses its own SPARQL-like query language (SKOSRec) to produce flexible on-the-fly recommendations. For identifying similar items the systems relies on SKOS concepts, but can be extended to other URI resources from the LOD cloud. The system uses Apache Jena[4] and can be applied on local as well remote SPARQL endpoints. The following section summarizes the general workflow of the SKOSRec system.

[4] https://jena.apache.org/.

1. **Parsing:** Parse the SKOSRec query.
2. **Compiling:** Decompose the query into the preferred input resource r (e.g. a movie) and a SPARQL graph pattern P.
3. **Resource retrieval:** Retrieve *relevant resources* from *SKOS annotations* of r in conjunction with P.
4. **Similar resources:** Score and rank the resources according to their *conditional similarity* with the input resource p.
5. **Recommendation:** Output the final recommendation results.

In the following, we will now rigorously define keywords in italics by using the notation for SPARQL semantics that was introduced by [25].

Definition 1 (SKOS annotations). *Let AG be the annotation graph of an RDF dataset D (AG ⊂ D), where resources are directly linked to concepts c of a SKOS system via a predefined property (e.g. dct:subject). All nodes of the AG are IRIs and the annotations of an input resource r are defined as follows:*

$$annot(r) = \{c \in AG \mid \exists < r, subject, c >\} \tag{1}$$

Upon retrieval of input resource annotations, similarity calculation does not have to be performed on the whole item space.

Definition 2 (Relevant resources). *The mapping Ω of relevant resources and their annotations is obtained by retrieving all resources P_r that share at least one SKOS concept with resource r. In addition to that, relevant resources are joined with a SPARQL graph pattern P, so that resources are excluded when certain user requirements are defined.*

$$P_r = (r, subject, ?c)\ AND\ (?x, subject, ?c) \tag{2}$$

$$\Omega = \{\mu(?x) | \mu \in [\![P]\!]_D\} \bowtie [\![P_r]\!]_{AG} \tag{3}$$

After querying all relevant resources and their annotations, similarity values can be calculated. They are based on the Information Content (IC) of the shared features of two resources. This idea was introduced by [23], but is expanded to the case when the item space is restricted to match a user defined graph pattern.

Definition 3 (Conditional similarity). *Let $annot(r)$ be the set of SKOS features of resource r and $annot(q)$ the set of SKOS features of resource q and $q \in \{\mu(?x) | \mu \in \Omega\}$, then their similarity can be derived from the IC of their shared concepts $C = \{annot(r) \cap annot(q)\}$*

$$sim(r, q) = IC_{cond}(C) \tag{4}$$

Definition 4 (Conditional Information Content). *The IC of a set of SKOS annotations is defined through the sum of the IC of each concept c ∈*

$\{\mu(?c)|\mu \in \Omega\}$, *where* $freq(c)$ *is the frequency of* c *among all relevant resources and* n *is the maximum frequency among these resources.*

$$IC_{cond}(C) = -\sum_{c \in C} log\left(\frac{freq(c)}{n}\right) \tag{5}$$

The retrieval of relevant resources and concept annotations can lead to long processing times, especially in cases when concepts are frequently used in a dataset. Hence, the number of records from SPARQL endpoints should be reduced. By knowing the length of the top-n recommendation list, it can be calculated which resources can be omitted without influencing the final ranking. This is the case when the maximum potential score for a certain number of shared features is smaller than the minimum potential score for a higher number of shared features. By this means, it is determined how many annotations have to be shared at least with an input resource (cut value) (see Eqs. 6 and 7).

$$\Omega_{cut} = \{\mu_{cut}(?x)|\mu_{cut} \in F_{count(?c)}(\Omega) > cut\} \tag{6}$$

$$\Omega_{reduced} = \Omega \bowtie \Omega_{cut} \tag{7}$$

4.2 Expressive SPARQL Integration

In the course of this paper, we are only able to give a short overview of the SKOSRec query language. Central to the idea of customizable on-the-fly recommendations is that both item similarity computation and querying of LOD resources can be flexibly integrated in a single query language. Even though there already exist some language extensions that combine SPARQL with imprecise parts [16,17], they do not take full advantage of the expressiveness of the RDF data model. Hence, we propose the SKOSRec query language that extends elements of the SPARQL 1.1 syntax (see underlined clauses in Listing 1) [26]. It enables flexible and powerful combinations of graph pattern matchings and subquerying with recommendation results. The 'RECOMMEND' operator issues the process of similarity calculation based on the input resource and potential user defined graph patterns, whereas the 'AGG' construct ensures that certain resources are exluded from the result set.

Listing 1. Grammar of SKOSRec

```
RecQuery        ::= Prologue SelectPart? SimProjection
                    Aggregation? ItemPart*
SelectPart      ::= SelectQuery
SimProjection   ::= RECOMMEND Var ItemLimit
Aggregation     ::= AGG IRIref
ItemLimit       ::= TOP INTEGER
ItemPart        ::= PREF (DECIMAL)? (VarPart | IRIPart)
IRIPart         ::= IRIref  (ConceptSim)?  (WherePart)?
VarPart         ::= Var (ConceptSim)? (WherePart)
ConceptSim      ::= C-SIM Relation DECIMAL
Relation        ::= ( < | > | <= | >= | = )
WherePart       ::= BASED ON WhereClause()
```

The central contributions of the new language are summarized below.

- **Recommendations for an input profile:** Whereas recommendations can be generated from both user and item data [16], we argue that the integration of local customer information and LOD resources is not feasible. An e-commerce retailer might avoid such a solution because of privacy concerns and additional costs and would rather prefer to obtain recommendations from outsourced repositories.
- **Graph pattern matching for preference information:** The SKOSRec language allows expression of preferences for variables that are contained in graph patterns. Thus, users can formulate vague preferences.
- **Subquerying with recommendation results:** In some areas it might be helpful to reuse recommendation results as a SPARQL-like subquery. Thus, triple stores can be powerfully navigated.

5 Experiments

We conducted several experiments to evaluate the viability of our approach. The goal of the evaluation was to find out, whether it is possible to get meaningful recommendation results with highly expressive queries from existing LOD repositories at reasonable computational cost. For this purpose, we issued SKOSRec queries from different target domains (movies, music, books and travel) (Table 1) to a local virtuoso server containing the DBpedia 3.9 dataset.

5.1 Scalable On-the-Fly Recommendations

The effectiveness of the optimization approach presented in Sect. 4.1 (Eqs. 6 and 7) was examined on 4 different datasets, where each dataset comprised 100 randomly selected DBpedia resources from the target domains. The performance test was carried out on an Intel Core i5 2500, clocked at 3.30 GHz with 8 GB of RAM. Evaluation results showed that, even though our approach imposes overhead that leads to slightly increased computational cost for smaller datasets (e. g. Fig. 4), it considerably reduces processing times for a growing number of triple statements for bigger datasets (Table 2, Figs. 1, 2 and 3).

Table 1. Overview of the target domains

	Movie	Book	Music	Travel
`rdf:type`	`dbo:Film`	`dbo:Book`	`Schema:MusicGroup`	`dbo:Place`
# items	90,063	31,172	86,013	725,546
# annotions p. item (∅)	7.207	4.410	6.060	2.422

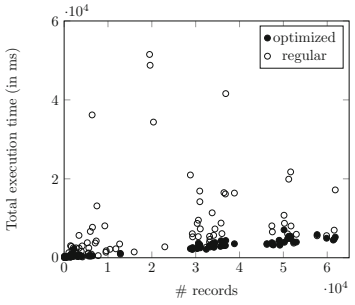

Fig. 1. Movie domain

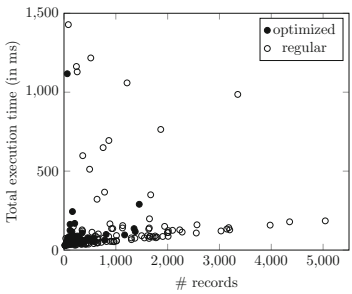

Fig. 2. Book domain

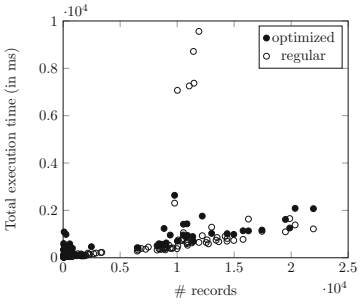

Fig. 3. Music domain

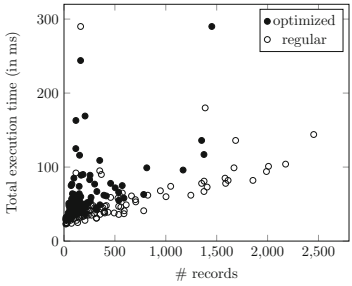

Fig. 4. Travel domain

Table 2. Results of the performance test

Domain	Exec. time in ms (∅)		# records (∅)	
	Regular	Optimized	Regular	Optimized
Movie	9,961	1,921	23,542	20,535
Book	303	92	1,206	837
Music	4,145	501	7,662	501
Place	218	73	621	134

5.2 Expressive SPARQL Integration

As former research on LDRS has already shown that the application of SKOS annotations leads to good precision and recall values in comparison to standard RS algorithms [9,10], we followed an explorative approach. We investigated, whether it is possible to formulate highly expressive SKOSRec queries that produce meaningful recommendation results from the DBpedia dataset. We issued advanced queries to showcase the viability of our language in several usage scenarios of the target domains.

Conditional Recommendations. This query template generates highly individual or business relevant recommendations. In the example depicted below, a marketer wants to obtain query results that are personalized and promote Christmas movies at the same time.

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

RECOMMEND ?movie TOP 3
PREF <r1> <r2>
WHERE {
 ?movie dct:subject ?c .
 ?c skos:broader* dbc:Christmas_films .
}
```

Input (r1, r2)	Output
The Devil Wears Prada Bridget Jones's Diary	Love Actually The Family Stone Scrooge
The Terminator Raiders of the Lost Ark	Ben-Hur Die Hard Trancers

Aggregation-Based Recommendations: Roll Up. When user preferences can be derived from their sublevel entities, the roll-up template might improve recommendations. Think of a travel agency intending to suggest city trips. Two customers would receive similar trip recommendations once they have been to the same cities even though they have visited different points of interest (POI). The example shows that it can be reasonable to instantiate the process of similarity calculation on sublevel entities to better fit recommendations to customer needs.

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

SELECT DISTINCT ?place (count(?place) as ?count)
WHERE {
 ?sight ?locatedIn ?place .
 ?place rdf:type yago:City108524735 .
}
GROUP BY ?place
ORDER BY DESC(COUNT(?place))
LIMIT 5
RECOMMEND ?sight TOP 1000
AGG <http://dbpedia.org/resource/Berlin>
PREF <r1> ... <r5>
```

Input (r1 ... r5)	Output
Checkpoint Charlie East Side Gallery Berlin Wall DDR Museum Stasi Museum	Moskau East Berlin Hamburg Trieste Warschau
Re:publica Berghain Friedrichshain E-Werk Bauhaus Archive	Vancouver London Amsterdam Paris Montreal

Aggregation-Based Recommendations: Drill Down. Sometimes customers find it hard to concretize their preferences. They might have a vague understanding of what they like, but could not tell why. In this example a user knows that somehow he/she likes movies directed by Quentin Tarantino. A drill-down query would find the most similar films to those that were directed by him and aggregate the results, such that related directors and their movies would be recommended.

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

SELECT DISTINCT ?director ?movie WHERE {
 ?movie dbo:director ?director .
}
RECOMMEND ?movie TOP 3
AGG dbr:Quentin_Tarantino
PREF ?prefMovie BASED ON {
 ?prefMovie dbo:director dbr:Quentin_Tarantino
}
```

?director	?movie
Robert Rodriguez	From Dusk till Dawn
Frank Miller	Sin City
Robert Rodriguez	Sin City
Tony Scott	True Romance

Cross-Domain Recommendations. Even though, standard collaborative filtering algorithms sometimes generate recommendations that are from a different domain than the items in a user profile, marketers cannot directly control the outputs. In contrast to that, the SKOSRec language enables explicit cross-domain querying. The following example shows that suggestions for novels (e. g. Beat novels) can be obtained by examining the user preference for a music group (e. g. The Beatles).

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

SELECT ?book (COUNT(?book) as ?count) WHERE {
 ?book dct:subject ?c .
 ?c skos:broader{,2} dbc:Novels . {
  SELECT ?book  WHERE {
   ?book ?p1 ?o . ?o ?p2 ?band . ?book rdf:type dbo:Book . }}}
GROUP BY ?book
ORDER BY DESC(COUNT(?book))
RECOMMEND TOP 10
PREF dbr:The_Beatles
```

Output (?book)
On the road
One Flew Over the Cuckoo's Nest
Sometimes a Great Notion

6 Conclusion

This paper demonstrated how Linked Open Data technologies can be utilized for highly flexible on-the-fly recommendations in e-commerce applications. Former LDRS calculated user preference predictions offline and thus prevented customizations and frequent updates of data sources as well as recommendation models. Although there have been efforts to enable user restrictions at runtime in query-based recommender systems, most of them either do not scale to large item spaces or do not handle data sparsity issues that arise when restricting the set of potential products to certain criteria. Above that, existing query-based LDRS do not yet take full advantage of the expressiveness that RDF and SPARQL graph patterns offer.

The SKOSRec prototype addresses this research gap by offering a powerful combination of similar resource retrieval and SPARQL graph pattern matchings from Linked Open Data repositories. Thus, individual and/or business preferences can be flexibly integrated. With the SKOSRec query language at hand, e-commerce retailers could define various recommendation workflows that can be adapted to specific usage contexts. For instance, the marketing department could use campaign templates and end users could enter their preferences through a user interface.

References

1. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans. Knowl. Data Eng. **17**(6), 734–749 (2005)
2. Schafer, J., Konstan, J., Riedl, J.: E-commerce recommendation applications. Appl. Data Min. Electron. Commer. **5**(1), 115–153 (2001)
3. Konstan, J., Miller, B., Maltz, D., Herlocker, J., Gordon, L., Riedl, J.: GroupLens - applying collaborative filtering to Usenet news. Commun. ACM **40**(3), 77–87 (1997)
4. Terveen, L., Hill, W., Amento, B., McDonald, D., Creter, J.: PHOAKS - a system for sharing recommendations. Commun. ACM **40**(3), 59–62 (1997)
5. Shardanand, U., Maes, P.: Social information filtering: algorithms for automating word of mouth. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 210–217 (1995)
6. Balabanovic, M., Shoham, Y.: Fab - content-based, collaborative recommendation. Commun. ACM **40**(3), 66–72 (1997)
7. Mobasher, B., Jin, X., Zhou, Y.: Semantically enhanced collaborative filtering on the web. In: Berendt, B., Hotho, A., Mladenič, D., van Someren, M., Spiliopoulou, M., Stumme, G. (eds.) EWMF 2003. LNCS (LNAI), vol. 3209, pp. 57–76. Springer, Heidelberg (2004)
8. Schmachterberg, M., Bizer, C., Paulheim, H.: State of the LOD Cloud 2014. http://linkeddatacatalog.dws.informatik.uni-mannheim.de/state/
9. Di Noia, T., Mirizzi, R., Ostuni, V., Romito, D.: Exploiting the web of data in model-based recommender systems. In: 6th ACM Conference on Recommender Systems, pp. 253–256 (2012)

10. Di Noia, T., Mirizzi, R., Ostuni, V., Romito, D., Zanker, M.: Linked open data to support content-based recommender systems. In: 8th International Conference on Semantic Systems, pp. 1–8 (2012)
11. Peska, L., Vojtas, P.: Enhancing recommender system with linked open data. In: Larsen, H.L., Martin-Bautista, M.J., Vila, M.A., Andreasen, T., Christiansen, H. (eds.) FQAS 2013. LNCS, vol. 8132, pp. 483–494. Springer, Heidelberg (2013)
12. Harispe, S., Ranwez, S., Janaqi, S., Montmain, J.: Semantic measures based on RDF projections: application to content-based recommendation systems. In: Meersman, R., Panetto, H., Dillon, T., Eder, J., Bellahsene, Z., Ritter, N., De Leenheer, P., Dou, D. (eds.) ODBASE 2013. LNCS, vol. 8185, pp. 606–615. Springer, Heidelberg (2013)
13. Koutrika, G., Bercovitz, B., Garcia-Molina, H.: FlexRecs - expressing and combining flexible recommendations. In: ACM SIGMOD International Conference on Management of Data, pp. 745–758 (2009)
14. Adomavicius, G., Tuzhilin, A.: Multidimensional recommender systems: a data warehousing approach. In: Fiege, L., Mühl, G., Wilhelm, U.G. (eds.) WELCOM 2001. LNCS, vol. 2232, pp. 180–192. Springer, Heidelberg (2001)
15. Adomavicius, G., Tuzhilin, A., Zheng, R.: REQUEST: a query language for customizing recommendations. Inf. Syst. Res. **22**(1), 99–117 (2011)
16. Ayala, A., Przyjaciel-Zablocki, M., Hornung, T., Schätzle, A., Lausen, G.: Extending SPARQL for recommendations. In: Semantic Web Information Management, pp. 1–8 (2014)
17. Kiefer, C., Bernstein, A., Stocker, M.: The fundamentals of iSPARQL: a virtual triple approach for similarity-based semantic web tasks. In: Aberer, K., et al. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 295–309. Springer, Heidelberg (2007)
18. Rosati, J., Di Noia, T., Lukasiewicz, T., Leone, R., Maurino, A.: Preference queries with Ceteris Paribus semantics for linked data. In: Debruyne, C., Panetto, H., Meersman, R., Dillon, T., Weichhart, G., An, Y., Ardagna, C.A. (eds.) OTM 2015. LNCS, vol. 9415, pp. 423–442. Springer, Sierre (2015)
19. Siberski, W., Pan, J.Z., Thaden, U.: Querying the semantic web with preferences. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 612–624. Springer, Heidelberg (2006)
20. Felfernig, A., Gordea, S., Jannach, D., Teppan, E., Zanker, M.: A short survey of recommendation technologies in travel and tourism. OEGAI J. **25**(7), 17–22 (2007)
21. Marie, N., Gandon, F., Ribiere, M., Rodio, F.: Discovery hub: on-the-fly linked data exploratory search. In: 9th International Conference on Semantic Systems, pp. 17–24 (2013)
22. Khrouf, H., Troncy, R.: Hybrid event recommendation using linked data and user diversity. In: 7th ACM Conference on Recommender Systems, pp. 185–192 (2013)
23. Meymandpour, R., Davis, J.: Recommendations using linked data. In: 5th Ph.D. Workshop on Information and Knowledge, pp. 75–82 (2012)
24. Ostuni, V., Di Noia, T., Di Sciascio, E., Mirizzi, R.: Top-n recommendations from implicit feedback leveraging linked open data. In: 7th ACM Conference on Recommender systems, pp. 85–92 (2013)
25. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of SPARQL. ACM Trans. Database Syst. TODS **34**(3), 16 (2009)
26. SPARQL 1.1 query language. https://www.w3.org/TR/sparql11-query/