Recommender Systems and Linked Open Data

Tommaso Di Noia[1](✉) and Vito Claudio Ostuni[2]

[1] SisInf Lab, Polytechnic University of Bari, Via Orabona 4, 70125 Bari, Italy
tommaso.dinoia@poliba.it
[2] Pandora Media Inc., 2101 Webster Street, Oakland, CA 9461, USA
vostuni@pandora.com

Abstract. The World Wide Web is moving from a Web of hyper-linked documents to a Web of linked data. Thanks to the Semantic Web technological stack and to the more recent `Linked Open Data` (`LOD`) initiative, a vast amount of RDF data have been published in freely accessible datasets connected with each other to form the so called `LOD` cloud. As of today, we have tons of RDF data available in the Web of Data, but only a few applications really exploit their potential power. The availability of such data is for sure an opportunity to feed personalized information access tools such as recommender systems. We present an overview on recommender systems and we sketch how to use `Linked Open Data` to build a new generation of semantics-aware recommendation engines.

1 Introduction

The recent emergence of social networks and pervasive mobile devices has contributed to the publication of a massive amount of information on the Web. We entered into an era of Information Overload: more information is produced than what we can really consume and process. Just to have an idea of what it means in practice, we know[1] that in just one minute about 694,445 searches are performed on Google, more than 6,600 pictures are uploaded on Flickr, about 13,000 hours of music streaming is done by the personalized Internet radio provider Pandora and so on.

Potentially, such enormous and heterogeneous collection of information allows users to find anything they may be looking for. However, in practice humans cannot process so much information without the assistance of any automatic filtering tool. Recommender Systems (RSs) [49] are a family of information filtering tools which have proven to be valuable means in assisting users to find, in a personalized manner, what is relevant for them in such overflowing complex information spaces. They provide users with personalized access to large collections of resources. On the one hand in the last twenty years we have assisted to the proliferation of this new kind of information filtering tools, namely recommender systems, which have proven to be very useful in supporting users in dealing with everyday decision making tasks in complex scenarios. Examples

[1] http://www.go-gulf.com/blog/60-seconds/.

W. Faber and A. Paschke (Eds.): Reasoning Web 2015, LNCS 9203, pp. 88–113, 2015.
DOI: 10.1007/978-3-319-21768-0_4

of such tasks are buying a product, looking for an accommodation or choosing the right movie to watch, just to cite few examples. On the other hand in the same temporal period we have also observed a shift from a Web conceived exclusively for humans to a Web of Data where information is made available also for machines.

Together with the appearing of social networks and Internet-enabled mobile devices, the Web has moved from a Web of hyper-linked Documents to one where both documents and data are linked. Thanks to the Semantic Web spread and to the more recent **Linked Open Data** (**LOD**) initiative, a vast amount of structured semantic data have been published in freely accessible datasets. More and more semantic data are published following the **Linked Data** [10] principles, that enable to set up links between objects in different data sources by connecting information in a single global data space: the **Web of Data**.

The matter in question is how to leverage the progresses made in the `LOD` field for improving that of recommender systems and vice versa. Here we see how the semantics encoded in the `Linked Open Data` can be used for improving traditional recommender systems. Actually, it is particularly interesting also to notice that such techniques can be used the other way around.

In this paper we introduce all the notions and elements needed to build and evaluate the effectiveness of a RS which leverages the data accessible in the `LOD` cloud. In the next section, we briefly review the recommendation problem and then in Sect. 3 we describe some basic metrics to evaluate the performance of a recommendation engine. In Sect. 4 we discuss on how to exploit the knowledge encoded in the `Linked Open Data` cloud to design a semantics-aware recommender system while in Sect. 5 we present some relevant related work.

2 Recommender Systems

Recommender Systems (**RSs**) are software tools and techniques providing suggestions for items to be of use to a user [49]. Such suggestions can relate to different decision-making processes, such as what users to connect to in a social network, what product to buy, what music to listen to, or what movie to watch. Products, music, movies are all examples of items in specific recommendation scenarios. Nowadays, almost every online service has a recommendation feature. Pandora[2], Netflix[3], Linkedin[4] and many others use recommendation functionalities in their systems to engage the users and offer them a better service.

The main aim of RSs is to help users in satisfying their information needs when dealing with huge information spaces. To achieve this, RSs try to select the subset of items which best match the users' preferences and tastes. Among the several definitions given in the literature, we report the one proposed by [15] which says: *the recommender system term indicates any system that produces individualized recommendations as output or has the effect of guiding the user*

[2] http://www.pandora.com/.
[3] http://www.netflix.com.
[4] http://www.linkedin.com.

Fig. 1. Example of Information Overload scenario.

in a personalized way to interesting or useful objects in a large space of possible options.

In Fig. 1 an example of typical Information Overload scenario is depicted where the user is exposed to a set of movies and does not know which one to select. If we contextualize this example to real situations where the user is overwhelmed with thousands/millions of items, then it is easy to imagine that it is very hard for her to make the right choice without any assistance.

In principle, the primary aim of both recommendation systems and search systems is to satisfy users' information needs. Nonetheless, there are quite a few fundamental differences between the two technologies. Towards the end of 2006, Jeffrey O'Brien, a Fortune writer, talking about recommender systems on the Web, quoted[5] *"The Web, they say, is leaving the era of search and entering one of discovery. What's the difference? Search is what you do when you're looking for something. Discovery is when something wonderful that you didn't know existed, or didn't know how to ask for, finds you"*. Compared to search systems, recommender systems provide the possibility for users to discover new resources that they may have not initially thought about, without the necessity of formulating their needs explicitly.

2.1 The Recommendation Problem

A formal formulation of the recommendation problem has been given in [2] and it is defined as follows. Let U represent the set of users and I the set of items in the system. Potentially, both sets can be very large. Let $f : U \times I \rightarrow R$, where R is a totally ordered set, be a utility function measuring the usefulness of item $i \in I$ for user $u \in U$. Then, the recommendation problem consists in finding for each user u such item $i^{max,u} \in I$ maximizing the utility function f. More formally, this corresponds to the following:

$$\forall u \in U,\ i^{max,u} = \arg\max_{i \in I}\ f(u, i)$$

[5] http://archive.fortune.com/magazines/fortune/fortune_archive/2006/11/27/8394347/index.htm.

Typically, the utility of an item is represented by a rating, which indicates how a particular user liked a particular item. The central problem of recommender systems is that the utility is not defined on the whole $U \times I$ space, but only a subset of it is actually available. For each user only a portion of her ratings is known. Hence, the main task of the system concerns the estimation of the utility function from the available data. Once the utility function is obtained it can be used to predict unknown values and recommendations are eventually generated by selecting for each user the best N items with highest utility (*top-N* recommendation list).

2.2 Users, Items and Ratings

As described in the formal definition of the recommendation problem, at the base of each RS there are three main essential elements which are: **users**, **items** and **ratings**. Usually such information are represented all together by means of a user-item ratings matrix. Such ratings matrix consists of a table where each row represents a user, each column represents a specific item, and each entry represents the rating given by the user to the particular item. Usually, such matrix results very sparse in practice because users rate only a small portion of items. Figure 2 shows an example of user-item ratings matrix in a movie RS where users express their preferences to the items (movies) by using a five points rating scale. The items with a question mark (unknown rating) are unseen for the corresponding user.

Users. Users are those actors of the system who are provided with recommendations. Users can be represented in different ways depending on the recommendation techniques used to compute recommendations. In order to provide personalized recommendations the system has to model and maintain information about their preferences. In a content-based RS users' preferences can be represented in a more transparent way by means of attribute/term vectors in a heuristic-based approach, by means of a model in a model-based approach or by means of knowledge representation tools (ontologies, rules, etc.).

Items. Item is the general term used to denote the resource the system recommends to users. Items may be characterized by their complexity and their value or utility [49]. Examples of items with low complexity and value are: news, Web pages, books, movies. While examples of more complex and higher value items can range from mobile phones, laptops to financial services, jobs and travels. Depending on the system and the recommendation technique the item content can be more or less structured and complex. It can range from just a numeric ID in a collaborative filtering system a to a a bag of keywords or set of attribute value pairs in a content-based system till to an ontology-based description in systems using a domain ontology.

Fig. 2. Example of user-item ratings matrix in a movie recommendation scenario.

Ratings. The most important thing RSs rely on is the availability of up to date information about users' preferences in the form of users' feedback. Depending on the way such information is collected, users' feedback can be classified as **explicit** or **implicit**. In the former case feedback come in the form of ratings. The user is asked to provide her opinion about an item on a rating scale which can be either numerical (e.g. 1–5 stars) or ordinal (strongly agree, agree, neutral, disagree, strongly disagree) or also binary (like/dislike). Although the explicit feedback case is more common in literature mostly due to the availability of many datasets with ratings, in practice is more common the case where the system gathers implicit feedback from the user. A system can infer the user preferences by monitoring user's behaviour without any bother to the user.

From Rating Prediction to Ranking. In the formulation of the recommendation problem given above the system is mainly seen as a predictive system in the way that the main goal is to accurately predict ratings. Such problem is known as the **rating prediction task**. However, the ultimate goal of the system in most situations is to provide the user with a ranked list of recommendations, namely **top-N recommendations**. As pointed out by [20] in many commercial systems, the *best bet* recommendations are shown, but the predicted rating values are not. This is usually referred to as a **top-N recommendation task**, where the goal of the recommender system is to find a few specific items which are supposed to be most appealing to the user. Other researchers [47] have refereed to such task also using a different terminology, namely **item recommendation task**, that is the task of predicting a personalized ranking on a set of items.

2.3 Recommendation Techniques

Depending on the the way the utility function is estimated and the availability of additional data about the characteristics of items for example, there are different types of recommendation techniques. The main two are: collaborative

filtering and content-based. Besides these two, there also other approaches such as knowledge-based, demographic and community-based just to cite a few. A complete list of techniques is given in [16] and in [49]. An important class of recommender systems which are often used in real systems are the hybrid recommenders [15] which combine different strategies to improve their separate performance and obtain higher recommendation quality.

Collaborative Filtering Recommendation. Collaborative Filtering is the process of filtering or evaluating items using the opinions of other people [52]. In this approach personalized recommendations for a target user are generated using opinions of users having similar tastes to those of the target user [48]. The main assumption in this approach is that users with similar preferences in the past will have similar preferences in the future.

Differently from any other technique the only input data that CF-RSs need is the user-item ratings matrix. Figure 3 shows a simple example of collaborative filtering case corresponding to the user-item ratings ratings matrix depicted in Fig. 2. If we consider `Alice` as target user, as said before, recommendations are generated considering the ratings given by other users with similar tastes. In this particular case, both `John` and `Alice` have similar tastes because they both rated similarly `Argo` and `Righteous Kill`. The system can exploit `John`'s ratings for estimating `Alice`'s unknown ratings. The basic intuition behind this method is that since `John` really likes `Heat` then also `Alice` may like it.

According to [12] there are two main types of collaborative filtering methods: memory-based and model-based. Memory-based CF uses a particular type of

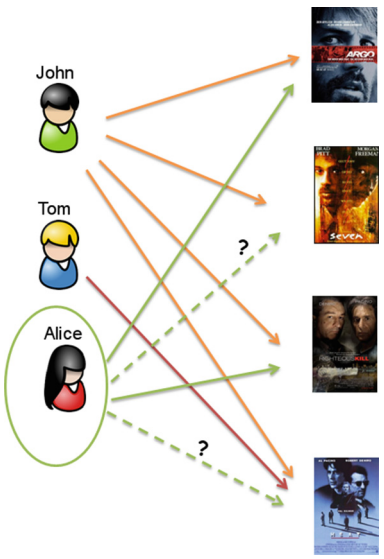

Fig. 3. Illustration of a CF-based recommender system.

Machine Learning methods that is the nearest neighborhood (k-NN) algorithm. The main property of such approach is that it does not require any preliminary model building phase because predictions are made by aggregating the ratings of the closest neighbours. On the contrary, model-based techniques first learn a predictive model which is eventually used to make predictions.

Memory-based approaches can be classified either in user-based or item-based. The user-based approach consists of predicting the relevance of an item for the target user by a linear combination of her neighbour's ratings, weighted by the similarity between the target user and such neighbours. One of the first implementation of such approach is the one presented in [48] which considers the rating deviations from the user's and neighbour's rating means ($\bar{r}_u$). Prediction for the active user u and target item i is computed as:

$$r_{u,i} = \bar{r}_u + \frac{\sum_{j=1}^{K}(r_{u_j,i} - \bar{r}_u) \cdot w_{u,u_j}}{\sum_{j=1}^{|U|} w_{u,u_j}}$$

where K is the number of neighbors for user u and w_{u,u_j} is the similarity weight between the active user u and neighbor u_j defined by the Pearson correlation coefficient:

$$w_{u,u_j} = \frac{\sum_i (r_{u,i} - \bar{r}_u) \cdot (r_{u_j,i} - \bar{r}_{u_j})}{\sqrt{\sum_{i=1}(r_{u,i} - \bar{r}_u)^2} \cdot \sqrt{\sum_{i=1}(r_{u_j,i} - \bar{r}_{u_j})^2}}$$

For a more detailed list of similarity measures and aggregation function please refer to [2]. The item-based CF approach bases on the usage of the same correlation-based or cosine-based techniques to compute similarities between items instead of users. The idea is to derive a notion of item similarity from user rating or purchase behavior and recommend items similar to those the user has already said they like. In [23] such idea has been applied to compute *top-N* item recommendations in e-commerce scenarios.

While at the beginning most of the research in this area focused on memory-based approaches, in the last years more attention has been paid to model-based techniques. In particular mode after the Netflix competition which showed that model-based techniques have higher accuracy [32]. The most adopted model-based approaches are the matrix factorization or latent factor models [33] which apply some form of dimensionality reduction on the user item ratings matrix to map both users and items into a joint lower dimensional latent factor space.

Even if collaborative filtering is the most widely adopted approach it can suffer from different drawbacks. First of all, to work properly it needs enough rating data to find meaningful correlations among items or users. This is main known as `sparsity` or `cold-start` problem [53]. In relation to that, there are two specific issues which are the `new user` and `new item problem`. When a new user enters the system till she has not rated a sufficient number of items the system is unable to compute reliable similarities with other users. When a new item is added to the catalog there is no way to recommend it before till no ratings about it are obtained. A typical way to tackle such cold-start problems is to

combine collaborative-filtering with content-based approaches. Another problem of CF is the so called `Grey sheep problem`, that is the inability of the system to properly treat users with very unusual preferences since the system is unable to find other similar users.

Content-Based Recommendation. Content-based RSs recommend an item to a user based upon a description of the item and a profile of the user's interests [46]. Briefly, the basic process performed by a content-based recommender consists in matching up the attributes of a user profile in which preferences and interests are stored, with the attributes of a content object (item) [36].

Differently from collaborative filtering, such recommendation approach relies on the availability of content features describing the items. Such features can be extracted from unstructured or semi-structured item descriptions by using proper Natural Language Processing (NLP) techniques or can be obtained from structured data as the case of tabular data in a relational database. A high level architecture of a content-based RS is presented in [36] (Fig. 5).

Figure 4 shows an example of content-based approach with reference to the user `Alice`. As we can see, differently from the CF case in this approach movies are provided with attributes, such as actors, genres, etc. The other difference is that only the target user is considered in the recommendation process. The basic intuition behind this approach is that since `Alice` likes `Argo` she might like `Heat` because they both belong to the `Drama` genre.

There are two main content-based recommendation approaches: *heuristic-based* or *model-based*.

Approaches using heuristic functions have their roots in Information Retrieval and Information Filtering. Items are recommended based on a comparison between their content and a user profile. The idea is to represent both items and users using typical IR techniques [6], e.g. vectors of terms, and compute a match between their representations. The user profile consists in a vector of terms built from the analysis of the items liked by the user. A typical approach is to use the Vector Space Model (VSM) [5] where items and user profiles can be represented as weighted vectors computed using the *tf-idf* formula [5]. The match

Fig. 4. Illustration of a content-based RS.

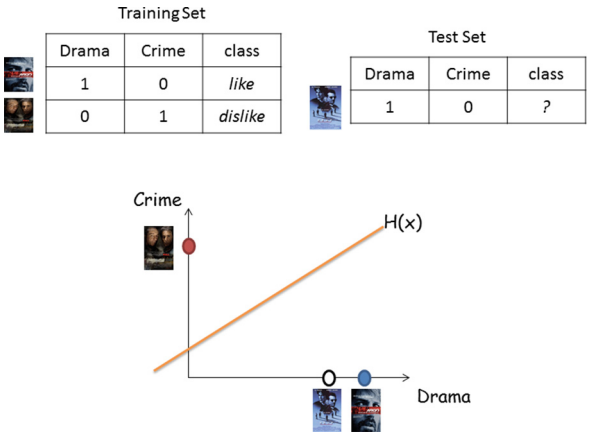

Fig. 5. Example of model-based CB-RS.

between items and user profile vectors can be computed using cosine similarity and eventually the most similar items to the user profile are recommended.

Model-based approaches [45] use Machine Learning techniques to learn a model of the user's preferences by analyzing the content characteristics of items the user rated. Specifically, a regression or classification model is learnt from a collection of items for which past user's ratings are available. The training set consists of item feature vectors labelled with ratings. Eventually, such learnt user model can be used for estimating the unknown ratings. This process is usually done for each user separately.

Differently from the heuristic-based case where the user model can be seen as an explicit representation of the user preferences (a vector containing the most preferred terms by the user), in this case the user profile is represented as a function obtained by means of an inductive learning process. Such function can be a complete black box or have a more interpretable form depending on the machine learning algorithm adopted.

A possible limitation of model-based approaches with respect heuristic-based ones is that the learning algorithm does not build a model with acceptable accuracy until it sees a relatively large number of examples (e.g. 50) [61].

Content-based methods can have several limitations. Maybe the main one is the `content overspecialization` which consists in the incapability of the system to recommend relevant items which are different to the ones the user already knows. Related to the previous issue, there is also the `portfolio effect` problem consisting in the redundancy and low diversity among the items in the recommendation lists.

Another limitation affecting CB systems is the `limited content analysis`. The quality of CB recommendations depends on the vailability and quality of features extracted from the items content.

For a complete and detailed description of content-based techniques for recommendations please refer to [36,46].

Knowledge-Based Recommandation. Both collaborative and content-based approaches work very well for all those domains where we have a user with an interaction history with the system. This is the case for instance of movies, books or music. Actually, there are some domains where it is quite difficult to have the user interacting with the system over the time. We may think of a student who wants to enroll at university or someone looking for a house to buy. In both cases it is unlikely that the users interact with the corresponding systems many times. Nevertheless, the information overload problem holds also in these situations and the help of a RS is highly desirable. A recommender system should guide the user through the set of possible choices by guessing or explicitly asking for her preferences. By combining its knowledge on the user desires and the one on the specific domain, the system selects a ranked list of items to be shown to the user. These classes of applications are classified as knowledge-based recommender systems [26].

Such systems are very often also referred to as *conversational* recommender systems [14]. Indeed, the user's preferences are elicited during her interaction with the system that may in turn ask her explicit questions regarding some characteristics of the item she is looking for. All these user requirements may vary in importance going from strict/hard to soft/graded requirements. Moreover, they can be updated while the user interplays with the application. Besides these user-generated constraints, as stated before, the system may also be aware of other constraint that are specific of the knowledge domain such as "`if` *the house has a big garden* `then` *it cannot be in the city center*".

We basically have two main types of knowledge-based recommender systems: *case-based* [13] and *constraint-based* [19,24,62] depending on the approach adopted in the representation and reasoning with user requirements and domain knowledge.

Hybrid Recommendation. The main idea behind hybrid recommender systems is to combine two or more classes of algorithms in order to mitigate the weaknesses of the individual approaches and obtain better recommendation quality. In [15] a taxonomy of several hybridization schemes is given which consists in the following list:

- Weighted: the scores provided by the individual recommenders are combined using a linear combination or a voting scheme;
- Switching: a special case of the previous type considering binary weights such that one recommender is turned on and the others are turned off;
- Mixed: recommendations generated from several recommenders are presented together at the same time by means of certain ranking or combination strategy;
- Feature combination: the features used by different recommenders are integrated and combined into a single data source, which is finally used by a single recommender;
- Cascade: the recommendation is performed as a sequential process where each recommender refines the recommendations given by the previous one;

– Feature augmentation: the output from one recommender is used as an additional input feature for other recommender;
– Meta-level: the model generated by one recommender is used as the input for another recommender.

Most common example of hybridization is the combination of collaborative and content strategies for mitigating CF limitations such as cold start and sparsity.

Semantics-Aware Recommendation. One of the main limitation of traditional content-based approaches is that they completely ignore the semantics associated to the item attributes because they rely on keyword-based representations. Keyword-based approaches to user profiling are unable to capture the semantics of user interests [22] because they are primarily driven by a string matching operation which suffers from problems of *polysemy*, the presence of multiple meanings for one term, and *synonymy*, multiple terms having the same meaning.

Furthermore, such textual approaches are incapable of capturing more complex relationships among objects at a deeper semantic level based on the inherent properties associated with these objects [21]. For example let us consider two generic movies *m1* and *m2*, which have *a1* and *a2* as directors, respectively. Let make the case that even if the two movies have different directors *a1* and *a2*, those directors have however many things in common such as they both were born in the same country and they both won a particular award. It is reasonable to assume that if a user likes *m1* because of *a1* then she might like with a certain degree *m2* because *a2* is similar to *a1*. In this case, an approach based on keyword matching would fail because the two values for the attribute director are different. When considering plain keyword representations possible relations among structured objects are completely missed. The system needs a better representation of the items content.

As described in [36] **semantic analysis** and its integration in personalization models is one of the most innovative and interesting approaches proposed in literature to solve those problems. The key idea is the adoption of knowledge bases for annotating items and representing profiles in order to obtain a *"semantic"* interpretation of the user information needs.

The core idea behind Semantics-aware Recommender Systems then, is to use ontological knowledge to describe items in order to have a deeper and more structured representation of their content.

The availability of additional semantic knowledge can allow the system to go beyond the simple keyword matching. Common-sense and domain-specific knowledge may be useful to give some meaning to the content of items, thus helping to generate more informative features than "plain" attributes [56]. Example of semantics-aware RSs can be any content-based or hybrid recommender where items are described by means of domain ontologies.

It is easy to see that depending on the addressed domain, we may build a semantics-aware RS that falls either in the content-based category or in the knowledge-based one.

3 Recommender Systems Evaluation

Several different recommendation approaches have been proposed in the last years. Generally, some of those different approaches can perform differently depending on the domain and on the task or other conditions such as sparseness of the ratings matrix. Clearly identifying the best algorithm for a given purpose has proven challenging, in part because researchers disagree on which attributes should be measured, and on which metrics should be used for each attribute [29]. Due to different reasons, the evaluation of recommender systems is inherently difficult to perform. For example different algorithms may be better or worse on different data sets or they may have different evaluation goals depending on the task. Furthermore, based on the adopted evaluation strategy, results may vary considerably.

An extensive review of evaluation metrics and techniques is provided in [29].

3.1 Metrics and Protocols

The most common aspect of recommendation quality measured in offline experimentations is **accuracy**. The literature on recommender systems typically distinguishes between two ways of measuring recommendation accuracy [59] which can be reconducted to two different main tasks which are rating prediction and ranking or *top-N* recommendations. Most of the evaluation methodologies adopted to asses the performances of recommendation systems are derived from the well established methodologies developed in the Information Retrieval field. This is particularly true when the system is used for *top-N* recommendation tasks. As reported in [7] although there are many commonalities between IR and recommendation systems there are also important differences to take into consideration. Two of the most significant ones regard the nature and the availability of relevance information about items. While in IR the relevance of a document with respect to a query is objective and is assessed by domain experts, in the RS field each user has her personal relevance for items which is determined by her ratings. Furthermore, in IR there is almost complete knowledge about such relevance information. This is not true at all for RSs because the system has knowledge only about a small portion of ratings for each user.

This latter aspect is crucial when evaluating ranking accuracy because some assumptions about the unknown ratings must be done. In [59] the authors argue that the main difference between the evaluation of the rating prediction and ranking tasks consists in how the training and test data are considered. They say that rating prediction is concerned with only the observed ratings, while ranking typically accounts for all items in the collection, whether the user has rated them or not. Hence, they present two protocols for evaluating ranking accuracy: **all unrated items** and **rated test-items**. The all unrated items protocol consists in creating a *top-N* recommendation list for each user by predicting a score for every item not rated by that particular user, whether the item appears in the user test set or not. Then, performance metrics are computed comparing recommendation lists with test data. The main assumption in this methodology

is that all the *unrated items* are considered to be *irrelevant* for the user with the effect of underestimating real recommendation quality. However, the authors of [59] argue that since the user-experience in *top-N* recommendation applications depends on the ranking of all items, this is a better evaluation methodology than the rated test-items one where only rated test items are considered for generating the *top-N* list. In fact, this latter method is the one adopted in evaluating the rating prediction task by using error based metrics.

Accuracy Metrics. Traditionally, the most popular metrics to measure the accuracy in the rating prediction task are error based metrics such as `Root Mean Squared Error` (`RMSE`) and `Mean Absolute Error` (`MAE`). The main goal in the rating prediction task is to predict the rating value that a user would assign to an item. Then the evaluation consists in predicting ratings $\hat{r}_{u,i}$ for a test set TS of user-item pairs (u, i) for which the true ratings $r_{u,i}$ are known.

$$\text{MAE} = \frac{1}{|TS|} \sum_{(u,i) \in TS} |\hat{r_{u,i}} - r_{u,i}| \tag{1}$$

$$\text{RMSE} = \sqrt{\frac{1}{|TS|} \sum_{(u,i) \in TS} (\hat{r_{u,i}} - r_{u,i})^2} \tag{2}$$

Such error based metrics can be useful for measuring rating prediction accuracy. Despite the large adoption of error metrics in the past several recent studies [9,20] have demonstrated that the accurate prediction of ratings does not imply the best *top-N* ranking of items. In case one wants to use such metrics for measuring the accuracy of *top-N* recommendations the main limitation of such metrics is that they do not make any distinction between the errors made on the high rated items and the errors made for the rest of the items.

More appropriate measures for evaluating *top-N* recommendation accuracy are precision-oriented metrics which take into account the ranked list of items. Examples of such metrics are `Precision`, `Recall` and `Normalized Discounted Cumulative Gain`. They are usually computed considering incremental list sizes, that is considering items up to a given ranking position (N). Typical values for N are 1, 5, 10, 25, 50, 100.

Precision and recall are binary metrics in the sense that they require binary rating data. Hence, we need to distinguish between relevant and not relevant items for the user. For example in a 5 points ratings scale, 4 and 5 ratings may be considered as relevant. In case of implicit feedback with unary rating data instead, all rated items can be considered as relevant.

Precision@N for user u ($P_u@n$) is computed as the fraction of *top-N* recommended items appearing in the user test set and which are relevant for the user, while Recall@N ($R_u@N$) is computed as the ratio of *top-N* recommended items appearing in the user test set which are also relevant to the number of relevant items in the user test set.

$$P_u@N = \frac{|L_u(N) \cap TS_u^+|}{n} \tag{3}$$

$$R_u@N = \frac{|L_u(N) \cap TS_u^+|}{|TS_u^+|} \tag{4}$$

where TS_u^+ is the set of relevant test items for u and $L_u(N)$ the ranked recommendation list up to position N. Both metrics are inversely related, typically an improvement in recall produces a decrease in precision.

Differently from precision and recall, the normalized discounted cumulative gain `nDCG` metric takes into account both relevance and rank position. Denoting with $r_{u,k}$ the rating given by user u to the item in position k in the *top-N* list, then `nDCG@N` for u can be defined as:

$$\text{nDCG@N} = \frac{1}{\text{IDCG@N}} \sum_{k=1}^{n} \frac{2^{r_{u,k}} - 1}{\log_2(1+k)} \tag{5}$$

where $IDCG@N$ indicates the score obtained by an ideal or perfect ranking of $L_u(N)$ and acts as normalization factor. When using the `all unrated items` protocol for those items with no rating in the test set a fixed default value can be assumed as suggested in [59].

Other Metrics. As pointed out by [39], the most accurate recommendations according to the standard metrics are sometimes not the recommendations that are most useful to users. Many researchers in the past proposed several metrics to measure the quality of the system from different perspectives. For example, an algorithm may provide very accurate recommendations but only for a small proportion of users or recommend only too popular items.

Some important qualities which have considered in literature besides accuracy regard the the ability of the system to compute *diverse* and *novel* suggestions. The novelty of a piece of information generally refers to how different it is with respect to *"what has been previously seen"*, by a specific user, or by a community as a whole. A possible way to compute recommendation novelty is to look at the popularity distribution of items. The `Entropy-Based Novelty` (`EBN`) [8] expresses the ability of a recommender system to suggest less popular items, i.e. items not known by a wide number of users. In particular, for each user's recommendation list $L_u(N)$, the novelty is computed as:

$$EBN_u@N = -\sum_{i \in L_u(N)} p_i \cdot \log_2 p_i$$

where:

$$p_i = \frac{|\{u \in U \mid i \text{ is relevant to } u\}|}{|U|}$$

In such formulation the lower $EBN_u@N$, the better the novelty. A broader discussion of novelty metrics is given in [60]. The aim of diversity metrics instead is to measure how diverse is the recommendation list. A well adopted diversity metric to measure the degree of diversification of the recommendation list is the `Intra-List Diversity` (`ILD`) [64].

Other important qualities of a system are catalog and user coverage. `User coverage` is the proportion of users to whom the system can recommend items. `Catalog coverage` is the percentage of the available items that are effectively recommended. A metric for measuring catalog coverage or equivalently, `aggregate diversity` [1], is the `diversity-in-top-N` metric presented in [1].

$$ADiv@N = \frac{|\bigcup_{u \in U} L_u(N)|}{|I|}$$

Low values of aggregated diversity indicate that all users are being recommended almost the same few items. This corresponds to a low level of personalization of the system.

4 Linked Open Data for Recommender Systems

Nowadays the Web of Data represents a huge repository of different kinds of knowledge spanning from sedimentary-one such as encyclopedic, linguistic, common-sense and so on, to real-time one such as data streams, events, etc. Several works on ontological or semantics-aware recommender systems have been proposed in the past before the `LOD` initiative was officially launched [3,17,22,40,41,55,56,63]. Most of them exploit item's ontological knowledge to boost collaborative filtering systems or to build better content-based ones. Such approaches have been shown to be particularly effective in solving some drawbacks of pure collaborative methods such as cold start and data sparsity, two well known problems in the recommender systems world. However, we argue that those approaches are not particularly suited for working with `LOD` datasets and new techniques are required for properly incorporating `LOD` into RSs and effectively exploiting their semantics.

We recognize two main reasons why new approaches are needed. The first reason is that those ontological recommendation algorithms developed before the `LOD` initiative referred principally to the usage of specific domain ontologies and taxonomies. `LOD` datasets have the peculiarity of being published according to the Semantic Web technologies and of using a graph-based data model. Such aspects require specific models and paradigms for their effective usage and incorporation into recommender systems.

Past works on ontology-based RSs base on the usage of taxonomies, controlled vocabulary and limited domain ontologies. With the advent of `LOD` new interesting possibilities appear for realizing better recommendation applications. The main advantages of using `LOD` for content-based and hybrid recommender systems can be summarized as:

- Availability of a great amount of multi-domain and ontological knowledge freely available for feeding the system;
- Semantic Web standards and technologies to retrieve the required data and hence no need for content analysis tasks for obtaining a structured representation of the items content;
- The ontological and relational nature of the data allows the system to analyze item descriptions at a semantic level.

Table 1. Datasets by domain.

Domain	Datasets
Government	183
Publications	96
Life sciences	83
User-generated content	48
Cross-domain	41
Media	22
Geographic	21
Social web	520

Multi-domain Knowledge. Depending on the dataset, there is the availability of multi-relational data related to different domains. We can get data about geographic locations, music, movies, art, people, facts, and general common-sense knowledge (see Table 1 [54]). If we consider encyclopedic datasets such as `DBpedia` [34] or `Freebase` [11], we have access to a huge amount of factual knowledge referring to a variety of topics. As pointed out by [56] factual and common sense knowledge bases can provide the system with the "cultural" background knowledge needed to compute an accurate content analysis. Another important advantage of datasets as `DBpedia` is their multi-lingual nature which grants the development of cross-language applications [43].

Standardized Access to Data. The usage of `Linked Open Data` datasets to retrieve information related to an item eases the pre-processing steps performed by the *Content Analyzer* [36] – the module of a CB-RS in charge of extracting relevant information from item descriptions – since the data is already structured in an ontological way and represented by using Semantic Web standards.

`LOD` datasets can be queried by means of their respective `SPARQL` endpoints. For `DBpedia`, it allows anyone to ask complex queries about any topic available in Wikipedia. For example, we can obtain information about which actors starred in the movie *Pulp Fiction* via a simple `SPARQL` query:

```
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
SELECT ?actor WHERE {
       dbpedia:Pulp_Fiction dbpedia-owl:starring ?actor.
}
```

Starting from the previous query we see how to extract rich data related to a specific resource/item as well as to a bunch of them. Given the `URI` corresponding to an item, it is possible for instance to extract the associated sub-graph by performing various `SPARQL` queries using a breadth-first search strategy up to a limited depth.

Semantic Analysis. The main advantage of using LOD is the availability of well structured graph-based item descriptions. In fact, items are connected to entities by means of semantic relations. Such entities are classified in more or less complex classes. The semantics of those classes and relations is described by means of ontologies. For example if we consider the resource dbpedia:Bruce_Willis in DBpedia, it is instance of the class dbpedia-owl:Person which in turn is sub-class of dbpedia-owl:Agent. In such ontology it is also defined the semantics of properties. For example the property dbpedia-owl:starring which connects dbpedia:Pulp_Fiction to dbpedia:Bruce_Willis has domain dbpedia-owl:Work and range dbpedia-owl:Actor which is sub-class of dbpedia-owl:Person.

Thanks to the semantic relations among entities the system can perform a deeper semantic analysis of the item content. In a keyword-based representation the system is limited to compute the syntactic match between keywords. Instead, thanks to the availability of semantic entities the system can potentially detect complex associations between the user profile and the items.

4.1 Feeding Recommender Systems with LOD

There are several aspects to consider in order to effectively incorporate LOD in recommendation applications. Ultimately, the goal is to provide the system with background knowledge about the domain of interest in the form of a knowledge graph. In Fig. 6 we show a high level architecture of a component in charge of retrieving portions of the LOD graph regarding the items in the system which are used to form the knowledge graph. Such component consists of two main modules: the Item Linker and the Item Graph Analyzer.

Item Linker. The Item Linker addresses the task of linking the items in the system with the corresponding resources in the LOD knowledge bases. The aim of such component is bridging the gap between between the items in the catalog and LOD. We have hypothesized two main ways for performing the linking task: Direct Item Linking, Item Description Linking. This module takes as input any dataset in the LOD cloud and the list of items in the catalog with associated side information, if available, and returns either the mapping between items and URIs or the set of URIs found in each item description, depending on the way the task is performed.

Direct Item Linking. This approach is the more straightforward way for accessing LOD datasets. However, it requires that items have to be LOD resources, otherwise it cannot be used.

Item Description Linking. This approach bases on the exploitation of side information about the items such as textual descriptions or attributes. Such information can be used as input for entity linking tools in order to have access to LOD resources and link them to the item. Specifically, entity linking is the task

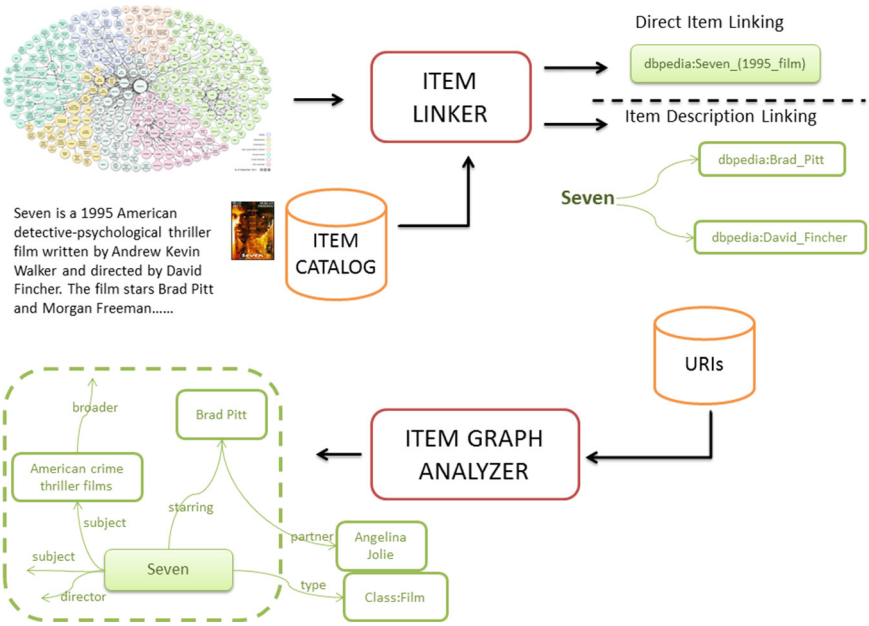

Fig. 6. High-level architecture for feeding RSs with LOD.

of linking the entity mentioned in the text with the corresponding real world entity in the existing knowledge base [57].

Item Graph Analyzer. This module is responsible of the extraction from the knowledge base of a descriptive and informative subgraph for each item, that is a set of `RDF` triples somehow related to the item resource. Eventually, all the extracted portions of `LOD` can be merged to obtain a specific knowledge graph representative of the domain of interest covered by the recommender. It takes as input the list of items URI returned by the Item Linker and returns a set of `RDF` triples for each item.

Performing some `SPARQL` queries for obtaining a set of `RDF` triples related to the item is an easy task, however extracting an informative and compact subgraph descriptive of the item is not. Potentially, each item resource may be connected to a big portion of the `LOD` graph. However, not all entities and relations may be informative and descriptive of the item content. Moreover, too much information may be problematic to handle.

After all, the main advantage of using `LOD` is that data are structured in an ontological way. Hence, one can consider specific classes and/or properties for extracting the subgraph of interest. The problem is how to use such information about classes and properties. Some properties can be very useful for a particular task and not for others. For example, the `dbpedia-owl:country` property can be useful in a location-based service, but maybe it is not in a movie recommen-

dation system. Speaking about classes, in a single domain recommender, class information is not that informative. For example in a movie recommender we can omit to consider the Movie class as feature because it would represent redundant information (all items – movies – are instances of the same class). Conversely, in a cross-domain system classes and relations among them may be very useful.

Several strategies to select a relevant subset of RDF triples for each item may be considered and adopted. One strategy can be to manually define a set of properties or sequences of properties by using some domain knowledge. One can automatically obtain a set of object properties related to the domain of interest by performing a SPARQL query like the following:

```
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
SELECT distinct(?p) where{
     ?s ?p ?o.
     ?s rdf:type dbpedia-owl:Film.
     ?p rdf:type owl:ObjectProperty.
}
```

4.2 Which Classes of RSs?

Due to the very rich and structured knowledge layer represented in the data available in the LOD cloud we may think to build different classes of recommender systems with respect to the ones introduced in Sect. 2. In particular we are allowed to build:

- *heuristic-based* content-based recommender systems;
- *model-based* content-based recommender systems;
- *hybrid* recommender systems;
- *knowledge-based* recommender systems.

4.3 Evaluating LOD-based RSs

There are many datasets available for the evaluation of recommender systems. However, such datasets are not appropriate for evaluating LOD-based recommendation algorithms because they do not contains links to URIs. In order to evaluate LOD-based RSs we can use three datasets belonging to three different domains which are movies, music and book. These datasets contain mappings between items (movies, artists, books) and their corresponding DBpedia URIs. The mappings for the datasets is available at http://sisinflab.poliba.it/semanticweb/lod/recsys/datasets/. In the following we describe the main characteristics of the three datasets.

Movielens. This dataset is based on the `MovieLens` 1M dataset[6] released by the GroupLens research group. The original dataset contains 1,000,209 1–5 stars ratings given by 6,040 users to 3,883 movies. We found a valid mapping for 3,148 out of the all movies.

LibraryThing. The second dataset is derived from the `LibraryThing`[7] dataset[8]. This dataset is related to the book domain and contains 7,112 users, 37,231 books and 626,000 ratings ranging from 1 to 10. In this case we found a match for 8,170 books.

LastFM. While the first two datasets contain explicit feedback data, this third dataset is based on implicit feedback consisting of user-artist listening data. This dataset comes from recent initiatives on information heterogeneity and fusion in recommender systems[9] [18]. It has been built on top of the `Last.fm` music system[10]. The original dataset contains 1,892 users, 17,632 artists and 92,834 relations between a user and a listened artist together with their corresponding listening counts. For this dataset we found a match for 9,490 out of a total of 17,632 artists.

5 Related Work

Many researches in the past have proposed different ways of using domain ontologies and taxonomies to improve the quality of conventional RSs.

In [37] the authors presented a content-based filtering approach wherein user and item profiles are described in terms of concepts belonging to a domain taxonomy. Specifically, the user profile is built by aggregating the concepts of items preferred by the user. The computation of the matching between user and item profiles base on a similarity function able to exploit the hierarchical taxonomy structure. They propose different possible matches between user and item such as exact or partial and different match scores depending also on the hierarchical distance between concepts. Such approach can be seen as a particular case of heuristic-based content recommendation technique where items are described using a domain taxonomy.

In [40] the authors describe an approach for ontological user profiling and an application of such approach for building a research paper recommendation system. In such system both research papers and user profiles are described in terms of topics organized in taxonomy. Each time the user browses a paper, the related topics are added to his profile together with the broader topics in the taxonomy. Those broader topics however just receive a smaller portion of the original topics.

[6] http://www.grouplens.org/node/73.
[7] http://www.librarything.com.
[8] http://www.macle.nl/tud/LT/.
[9] http://ir.ii.uam.es/hetrec2011/datasets.html.
[10] http://www.lastfm.com.

In this way also general topics were added to the user profile in order to have a deeper content representation. Recommendations were eventually computed considering the correlation between the user's topics of interest and papers classified to those topics. In [41] the authors present a semantically enhanced collaborative filtering approach where structured semantic knowledge about items is used in conjunction with user-item ratings to create a combined similarity measure for item comparisons. Taxonomic information is used in [63] to represents the user's interest in categories of products. Consequently, user similarity is determined by common interests in categories and not by common interests in items. In [3] the authors present an approach that infers user preferences from rating data using an item ontology. The system collaboratively generates recommendations using the ontology and infers preferences during similarity computation. Another hybrid ontological recommendation system is proposed in [17] where user preferences and item features are described by semantic concepts to obtain users' clusters corresponding to implicit *Communities of Interest*.

A semantic content-collaborative hybrid recommender is presented in [22] which computes similarities between users relying on their content-based profiles. The particularity of such work is the usage of sense-based user profiles instead of keyword-based ones. Such semantic profiles are obtained by integrating machine learning algorithms for text categorization with a word sense disambiguation strategy based exclusively on the lexical knowledge stored in WordNet. Most of the presented works used ontologies to compute better user-user or item-item similarities in memory-based collaborative filtering approaches. However little work has been done in exploiting ontologies for computing model-based recommendations. A detailed description of recommendation techniques based on ontological filtering is given in [30,36].

In the last few years with the availability of `Linked Open Data` a new class of recommender systems has emerged which can be named as `LOD`-based recommender systems. This new typology of recommendation methods is attracting increasingly interest in both the communities of Semantic Web and Recommender Systems.

Most of the proposed works regarding this topic tried to reuse and adapt some of the ideas presented in the context of ontological RSs to `LOD` datasets which have their own characteristics, while others proposed new approaches specifically suited for working with Linked Data technologies and others proposed new applications of recommendation technologies for Linked Data. In what follows we review the most significant contributions.

One of the first approaches that exploits `Linked Open Data` for building recommender systems is [28]. Here the authors, for the first time propose a recommender system fed by `Linked Open Data`. In [27] the authors present a knowledge-based framework leveraging `DBpedia` for computing cross-domain recommendations. In [35] the authors propose a graph-based recommendation approach utilizing model- and memory-based link prediction methods. In [38] `LOD` datasets are used for personalized exploratory search using a spreading activation method. They use a spreading activation method with the purpose of finding semantic relatedness between items belonging to different domains. *dbrec* [44] is a

music content-based recommender system leveraging the DBpedia dataset. They define the *Linked Data Semantic Distance* in order to find semantic distances between resources and then compute recommendations.

A full SPARQL-based recommendation engine named RecSPARQL is presented in [4]. The proposed tool extends the syntax and semantics of SPARQL to enable a generic and flexible way for collaborative filtering and content-based recommendations over arbitrary RDF graphs. The authors of [58] propose an approach for topic suggestions based on some proximity measures defined on the top of the DBpedia graph.

In [31] the authors present an event recommendation system based on linked data and user diversity. A semantic-aware extension of the SVD++ model, named SemanticSVD++, is presented in n [50]. It incorporates semantic categories of items into the model. The model is able also to consider the evolution over time of user's preferences. In [51] the authors improve their previous work for dealing with cold-start items by introducing a vertex kernel for getting knowledge about the unrated semantic categories starting from those categories which are known. Another interesting direction about the usage of LOD for content-based RSs is explored in [42] where the authors present Contextual eVSM, a content-based context-aware recommendation framework that adopts a semantic representation based on distributional models and entity linking techniques. In particular entity linking is used to detect entities in free text and map them to LOD.

Finally, in [25] the authors propose the usage of recommendation techniques for providing personalized access to Linked Data. The proposed recommendation method is a user-user collaborative filtering recommender wherein the similarity between the users takes into account the commonalities and informativeness of the resources instead of treating resources as plain identifiers.

References

1. Adomavicius, G., Kwon, Y.: Improving aggregate recommendation diversity using ranking-based techniques. IEEE Trans. Knowl. Data Eng. **24**(5), 896–911 (2012)
2. 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)
3. Anand, S.S., Kearney, P., Shapcott, M.: Generating semantically enriched user profiles for web personalization. ACM Trans. Internet Technol. **7**(4), October 2007
4. Ayala, V.A.A., Przyjaciel-Zablocki, M., Hornung, T., Schätzle, A., Lausen, G.: Extending SPARQL for recommendations. In: Proceedings of Semantic Web Information Management on Semantic Web Information Management, SWIM 2014, pp. 1:1–1:8. ACM, New York (2014)
5. Baeza-Yates, R.A., Ribeiro-Neto, B.: Modern Information Retrieval: The Concepts and Technology behind Search. Addison-Wesley Professional, Boston (2011)
6. Balabanović, M., Shoham, Y.: Fab: Content-based, collaborative recommendation. Commun. ACM **40**(3), 66–72 (1997)

7. Bellogín, A.: Performance prediction and evaluation in recommender systems: an information retrieval perspective. Ph.D. thesis, Escuela Politécnica Superior Departamento de Ingeniería Informática (2012)
8. Bellogín, A., Cantador, I., Castells, P.: A study of heterogeneity in recommendations for a social music service. In: Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems, HetRec 2010, pp. 1–8. ACM, New York (2010)
9. Bellogin, A., Castells, P., Cantador, I.: Precision-oriented evaluation of recommender systems: an algorithmic comparison. In: Proceedings of the Fifth ACM Conference on Recommender Systems, RecSys 2011, pp. 333–336. ACM, New York (2011)
10. Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far. Int. J. Semant. Web Inf. Syst **5**(3), 1–22 (2009)
11. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, pp. 1247–1250. ACM, New York (2008)
12. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, UAI 1998, pp. 43–52 (1998)
13. Bridge, D., Göker, M.H., McGinty, L., Smyth, B.: Case-based recommender systems. Knowl. Eng. Rev. **20**(3), 315–320 (2005)
14. Burke, R.: Knowledge-based recommender systems. In: Kent, A. (ed.) Encyclopedia of Library and Information Science, vol. 69, pp. 181–201. CRC Press, Boca Raton (2000)
15. Burke, R.D.: Hybrid recommender systems: survey and experiments. User Model. User-Adapt. Interact. **12**(4), 331–370 (2002)
16. Burke, R.: Hybrid web recommender systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) Adaptive Web 2007. LNCS, vol. 4321, pp. 377–408. Springer, Heidelberg (2007)
17. Cantador, I., Bellogín, A., Castells, P.: A multilayer ontology-based hybrid recommendation model. AI Commun. Special Issue Recomm. Syst. **21**(2—-3), 203–210 (2008)
18. Cantador, I., Brusilovsky, P., Kuflik, T.: 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011). In: Proceedings of the 5th ACM Conference on Recommender systems, RecSys 2011. ACM, New York (2011)
19. Colucci, S., Di Noia, T., Di Sciascio, E., Donini, F.M., Ragone, A.: Knowledge elicitation for query refinement in a semantic-enabled e-marketplace. In: Proceedings of the 7th International Conference on Electronic Commerce, ICEC 2005, pp. 685–691. ACM, New York (2005)
20. Cremonesi, P., Koren, Y., Turrin, R.: Performance of recommender algorithms on Top-N recommendation tasks. In: Proceedings of the Fourth ACM Conference on Recommender Systems, RecSys 2010, pp. 39–46. ACM, New York (2010)
21. Dai, H., Mobasher, B.: A road map to more effective web personalization: integrating domain knowledge with web usage mining. In: Proceedings of the International Conference on Internet Computing, IC 2003, Las Vegas, Nevada, USA, 23–26 June 2003, vol. 1, pp. 58–64 (2003)
22. Degemmis, M., Lops, P., Semeraro, G.: A content-collaborative recommender that exploits wordnet-based user profiles for neighborhood formation. User Model. User-Adapt. Inter. **17**(3), 217–255 (2007)

23. Deshpande, M., Karypis, G.: Item-based Top-N recommendation algorithms. ACM Trans. Inf. Syst. **22**(1), 143–177 (2004)
24. Di Noia, T., Di Sciascio, E., Donini, F.M.: Semantic matchmaking as non-monotonic reasoning: a description logic approach. J. Artif. Int. Res. **29**(1), 269–307 (2007)
25. Dojchinovski, M., Vitvar, T.: Personalised access to linked data. In: Janowicz, K., Schlobach, S., Lambrix, P., Hyvönen, E. (eds.) EKAW 2014. LNCS, vol. 8876, pp. 121–136. Springer, Heidelberg (2014)
26. Felfernig, A., Burke, R.: Constraint-based recommender systems: technologies and research issues. In: Proceedings of the 10th International Conference on Electronic Commerce, ICEC 2008, pp. 3:1–3:10. ACM, New York (2008)
27. Fernández-Tobías, I., Cantador, I., Kaminskas, M., Ricci, F.: A generic semantic-based framework for cross-domain recommendation. In: Proceedings of the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems, HetRec 2011, pp. 25–32. ACM, New York (2011)
28. Heitmann, B., Hayes, C.: Using linked data to build open, collaborative recommender systems. Linked data meets artificial intelligence. In: AAAI Spring Symposium (2010)
29. Herlocker, J.L., Konstan, J.A., Terveen, L.G., Riedl, J.T.: Evaluating collaborative filtering recommender systems. ACM Trans. Inf. Syst. **22**(1), 5–53 (2004)
30. Jannach, D., Zanker, M., Felfernig, A., Friedrich, G.: Recommender systems and the next-generation web. In: Recommender Systems, pp. 253–288. Cambridge University Press, Cambridge (2010)
31. Khrouf, H., Troncy, R.: Hybrid event recommendation using linked data and user diversity. In: Proceedings of the 7th ACM Conference on Recommender Systems, RecSys 2013, pp. 185–192. ACM, New York (2013)
32. Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2008, pp. 426–434. ACM, New York (2008)
33. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer **42**(8), 30–37 (2009)
34. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., Bizer, C.: DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. Semant. Web J. **6**(2), 167–195 (2015)
35. Lommatzsch, A., Plumbaum, T., Albayrak, S.: A linked dataverse knows better: boosting recommendation quality using semantic knowledge. In: Proceedings of the 5th International Conference on Advances in Semantic Processing, pp. 97–103. IARIA, Wilmington (2011)
36. Lops, P., Gemmis, M., Semeraro, G.: Content-based recommender systems: state of the art and trends. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P.B. (eds.) Recommender Systems Handbook, pp. 73–105. Springer, USA (2011)
37. Maidel, V., Shoval, P., Shapira, B., Taieb-Maimon, M.: Evaluation of an ontology-content based filtering method for a personalized newspaper. In: Proceedings of the 2008 ACM Conference on Recommender Systems, RecSys 2008, Lausanne, Switzerland, 23–25 October 2008, pp. 91–98 (2008)
38. Marie, N., Corby, O., Gandon, F., Ribière, M.: Composite interests' exploration thanks to on-the-fly linked data spreading activation. In: Proceedings of the 24th ACM Conference on Hypertext and Social Media, HT 2013, pp. 31–40. ACM, New York (2013)

39. McNee, S.M., Riedl, J., Konstan, J.A.: Being accurate is not enough: how accuracy metrics have hurt recommender systems. In: CHI 2006 Extended Abstracts on Human Factors in Computing Systems, CHI EA 2006, pp. 1097–1101. ACM, New York (2006)
40. Middleton, S.E., Shadbolt, N.R., De Roure, D.C.: Ontological user profiling in recommender systems. ACM Trans. Inf. Syst. **22**, 54–88 (2004)
41. 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)
42. Musto, C., Semeraro, G., Lops, P., de Gemmis, M.: Combining distributional semantics and entity linking for context-aware content-based recommendation. In: Dimitrova, V., Kuflik, T., Chin, D., Ricci, F., Dolog, P., Houben, G.-J. (eds.) UMAP 2014. LNCS, vol. 8538, pp. 381–392. Springer, Heidelberg (2014)
43. Narducci, F., Palmonari, M., Semeraro, G.: Cross-language semantic retrieval and linking of E-Gov services. In: Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N., Welty, C., Janowicz, K. (eds.) ISWC 2013, Part II. LNCS, vol. 8219, pp. 130–145. Springer, Heidelberg (2013)
44. Passant, A.: Measuring semantic distance on linking data and using it for resources recommendations. In: Proceedings of the AAAI Spring Symposium "Linked Data Meets Artificial Intelligence", vol. 3 (2010)
45. Pazzani, M., Billsus, D.: Learning and revising user profiles: the identification ofinteresting web sites. Mach. Learn. **27**(3), 313–331 (1997)
46. Pazzani, M.J., Billsus, D.: Content-based recommendation systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) Adaptive Web 2007. LNCS, vol. 4321, pp. 325–341. Springer, Heidelberg (2007)
47. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: bayesian personalized ranking from implicit feedback. In: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI 2009, pp. 452–461. AUAI Press, Arlington (2009)
48. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J.: Grouplens: an open architecture for collaborative filtering of netnews. In: CSCW 1994, Proceedings of the Conference on Computer Supported Cooperative Work, Chapel Hill, NC, USA, 22–26 October 1994, pp. 175–186 (1994)
49. Ricci, F., Rokach, L., Shapira, B., Kantor, P.B. (eds.): Recommender Systems Handbook. Springer, USA (2011)
50. Rowe, M.: SemanticSVD++: incorporating semantic taste evolution for predicting ratings. In: 2014 IEEE/WIC/ACM International Conferences on Web Intelligence, WI 2014 (2014)
51. Rowe, M.: Transferring semantic categories with vertex kernels: recommendations with SemanticSVD++. In: Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C., Vrandečić, D., Groth, P., Noy, N., Janowicz, K., Goble, C. (eds.) ISWC 2014, Part I. LNCS, vol. 8796, pp. 341–356. Springer, Heidelberg (2014)
52. Schafer, J.B., Frankowski, D., Herlocker, J., Sen, S.: Collaborative filtering recommender systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) Adaptive Web 2007. LNCS, vol. 4321, pp. 291–324. Springer, Heidelberg (2007)
53. Schein, A.I., Popescul, A., Ungar, L.H., Pennock, D.M.: Methods and metrics for cold-start recommendations. In: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2002, pp. 253–260. ACM, New York (2002)

54. Schmachtenberg, M., Bizer, C., Paulheim, H.: Adoption of the linked data best practices in different topical domains. In: Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C., Vrandečić, D., Groth, P., Noy, N., Janowicz, K., Goble, C. (eds.) ISWC 2014, Part I. LNCS, vol. 8796, pp. 245–260. Springer, Heidelberg (2014)
55. Semeraro, G., Degemmis, M., Lops, P., Basile, P.: Combining learning and word sense disambiguation for intelligent user profiling. In: IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, 6–12 January 2007, pp. 2856–2861 (2007)
56. Semeraro, G., Lops, P., Basile, P., de Gemmis, M.: Knowledge infusion into content-based recommender systems. In: Proceedings of the Third ACM Conference on Recommender Systems, RecSys 2009, pp. 301–304, ACM, New York (2009)
57. Shen, W., Wang, J., Luo, P., Wang, M.: Linden: linking named entities with knowledge base via semantic knowledge. In: Proceedings of the 21st International Conference on World Wide Web, WWW 2012, pp. 449–458. ACM, New York (2012)
58. Stankovic, M., Breitfuss, W., Laublet, P.: Linked-data based suggestion of relevant topics. In: Proceedings of the 7th International Conference on Semantic Systems, I-Semantics 2011, pp. 49–55. ACM, New York (2011)
59. Steck, H.: Evaluation of recommendations: rating-prediction and ranking. In: RecSys, pp. 213–220 (2013)
60. Vargas, S., Castells, P.: Rank and relevance in novelty and diversity metrics for recommender systems. In: Proceedings of the Fifth ACM Conference on Recommender Systems, RecSys 2011, pp. 109–116. ACM, New York (2011)
61. Webb, G.I., Pazzani, M.J., Billsus, D.: Machine learning for user modeling. User Model. User-Adap. Inter. **11**(1–2), 19–29 (2001)
62. Zanker, M., Jessenitschnig, M., Schmid, W.: Preference reasoning with soft constraints in constraint-based recommender systems. Constraints **15**(4), 574–595 (2010)
63. Ziegler, C.-N., Lausen, G., Schmidt-Thieme, L.: Taxonomy-driven computation of product recommendations. In: Proceedings of the Thirteenth ACM International Conference on Information And Knowledge Management, CIKM 2004, pp. 406–415. ACM, New York (2004)
64. Ziegler, C.-N., McNee, S.M., Konstan, J.A., Lausen, G.: Improving recommendation lists through topic diversification. In: Proceedings of the 14th International Conference on World Wide Web, WWW 2005, pp. 22–32. ACM, New York (2005)