Anatomy of a Semantic Web-enabled Knowledge-based Recommender System

Daniele Dell'Aglio, Irene Celino, and Dario Cerizza

CEFRIEL – Politecnico of Milano, Via Fucini 2, 20133 Milano, Italy
{name.surname}@cefriel.it

Abstract. Knowledge-based Recommender Systems suggest to users items of their interest, on the basis of some understanding of both items' characteristics and users' profiles. In order to properly work, this kind of recommender systems need a thorough modeling of items and users; the usual barrier to their development is, therefore, the availability of the necessary knowledge and its maintenance over time.
With this respect, Semantic Web technologies can be of great help: not only knowledge technologies and languages can be employed to build the knowledge base, but the large availability of open and linked data about a growing variety of fields and topics, published on the Web of Data, further simplifies the modeling step for recommender systems.
In this paper, we present our concept of Semantic Web-enabled Recommender System, based on the retrieval from the linked data Web of the necessary pieces of knowledge about items and users. We illustrate the general structure of this new family of Knowledge-based Recommender Systems and we explain how we concretely followed this approach to develop a tool to recommend Web services in the context of the SOA4All project. We also offer our considerations about the strengths, the current limitations and the possible extensions of our proposal.

1 Introduction and Motivation

Recommender systems are becoming more and more commonly used to help users serendipitously find items they were (implicitly or explicitly) looking for. From a user's point of view, recommendations are seen as suggestions that are proactively provided by the system, in a timely fashion. In order to be effectively useful, the recommendations should be accurate, as to "foresee" a user's needs.

Recommender systems are usually classified by the recommendation technique they use [9]:

- *Collaborative Filtering Recommender Systems* [30]: given a user, they find users with similar behavior to predict items of interest;
- *Content-based Recommender Systems* [25]: they usually employ a classifier to predict items' similarity;
- *Demographic Recommender Systems*: they compute users' similarity using demographic information (age, location, etc.);
- *Knowledge-based Recommender Systems* [8]: they build a knowledge base with a model of the users and/or items in order to apply inference techniques and find matches between users' need and items' features.

Additionally, another category of systems, the *Hybrid Recommender Systems* [9], tries to join the advantages of two or more techniques described above. In this paper we focus on the Knowledge-based Recommender Systems.

Knowledge-based Recommender Systems offers some advantages with respect to the other techniques. First, they need a *minimal amount of users*, i.e. they do not require a huge amount of data to compute recommendations, differently from other classes of Recommender Systems; moreover, they do not suffer the so-called *cold start problem*: when a new user/item is added with its description, the system is immediately able to compute recommendations for the new user/item; finally, by their very nature, they are able to generate *proofs for the recommendations*, i.e. they are able to "explain" the motivation behind an item proposal on the basis of the user/item modeling at disposal.

The main drawback of Knowledge-based Recommender Systems, however, consists in the modeling, building and maintenance of the knowledge base: a correct and up-to-date description of items to be recommended, as well as of users to which provide suggestions must be ensured, in order to guarantee a high level of recall and precision of the generated recommendations. Several elements can change over time: new users and new items can be added, old users and old items can require an updated description, the domain knowledge can evolve or be modified to take into consideration new features, the set of policies defined to compute recommendations can be revised to better meet users' needs and requirements, and so on.

This knowledge base creation and maintenance require a lot of effort, with a heavy human intervention. A special attention must also be given to assure the consistency, quality and reliability of the modeled knowledge. Therefore, in this context, it is quite natural to think about applying technologies and tools coming from the Semantic Web community to help and support this phase. Some efforts in this directions were already explored (as we report in Section 5); still, a comprehensive work to design a Semantic Web-enabled Knowledge-based Recommender System is missing.

In this paper, we present a holistic approach to apply Semantic Web technologies and the knowledge coming from the Web of Data to support and enhance the knowledge base modeling as well as all other phases of the recommender life cycle. Indeed, we believe that the recent availability of large amounts of linked data can definitely offer new opportunities to build innovative and enriched recommender systems. Moreover, the widespread uptake of Semantic Web standards – RDF [15], SPARQL [29] and also the recently published RIF [6] W3C Recommendation – provides new favorable circumstances to improve recommendation algorithms and tools, by leveraging on users' and items' semantics.

The rest of the paper is structured as follows: in Section 2 we describe our concept for a *Semantic Web-enabled Recommender System* and we introduce the scenario in which we applied our proposed approach to recommend Web services, in the context of the SOA4All project. The details of our approach are offered in Section 3, in which we explain how the Web of Data can be leveraged to build a knowledge base, and in Section 4, where we illustrate how Semantic

Web technologies can help in computing recommendations; those sections include also the description of the actual realization of our conceptual approach in the Web service recommendation scenario. Finally, in Section 5, we introduce some previous works to combine Semantic Web technologies with Recommender Systems and in Section 6 we conclude with our considerations about open issues and current limitations and we provide hints for future extensions.

2 Our Concept of Semantic Web-enabled Recommender System

For the last years, the Semantic Web community has been working to realize the *Web of Data*, i.e. to publish structured information and datasets on the Web and interlinking them. Thanks to the popularity of Tim Berners-Lee linked data meme [3] and to the growing interest and coordinated effort of the Web community, a first nucleus of this Web of Data has been built and constantly updated and enriched since early 2007 to constitute the so-called "Linking Open Data dataset cloud" or simply *LOD Cloud*[1], which comprises around 4.7 billion RDF statements, connected by 142 million RDF links (as of May 2009 [4]).

We believe that the Web of Data should be considered as an interesting source of information to be used by Knowledge-based Recommender Systems. The LOD Cloud is a huge public source where information can be found to describe several kinds of items, users and domains. Accessing and exploiting the Web of Data can allow the partial automation of the knowledge base creation and maintenance, simplifying the modeling and profiling of items and users. Furthermore, the computation tasks to generate recommendations, operating on the knowledge base, can be performed and enhanced by the use of Semantic Web tools like SPARQL processors, reasoners or rule-based systems.

In this paper we propose a holistic approach to design and to realize a Knowledge-based Recommender System using Semantic Web technologies. In Section 2.1 we explain the high level architecture of such a system, which will be detailed in Sections 3 and 4. The scenario in which we demonstrate the applicability of our approach is introduced in Section 2.2; we will use this application scenario also in the following sections to explain how we realized in practice a prototype of our Semantic Web-enabled Recommender System.

2.1 High-level Architecture of a Semantic Web-enabled Recommender System

A recommender system is usually part of a more complex application (like a Web site), so it exchanges data with other components. It provides its functionalities to a set of known users about a predefined set of items; both those sets can be modified over time (users and items can be removed or added).

Our concept of recommender system processes information from both private and public sources. We call *private data* the information about users and items generated by other parts of the application; with *public data* we identify the data published and freely accessible on the Web of Data.

[1] Cf. `http://lod-cloud.net/`.

Figure 1 shows the schematic high-level architecture of our Semantic Web-enabled Recommender System, that consists of two main components, the Model Builder and the Recommendation Engine.

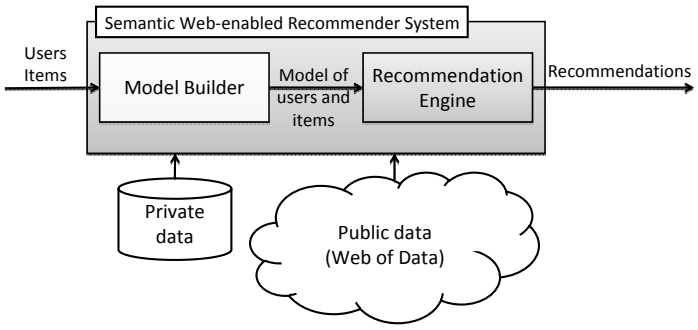

Fig. 1. Schema of our Semantic Web-enabled Recommender System

The *Model Builder* is related to the knowledge base building and maintenance; its goal is to create a model containing the descriptions of items' and users' profiles. Additionally, this model should contain information about how users (and their interests) are related to items (and their features). In order to build such a model, this component queries and interacts with the available data sources, both private and public ones, and identifies the knowledge "bits" useful to describe the items to be recommended and the users. For what regards the items, the Model Builder should look for available descriptions, categorizations and classifications, reviews and ratings, related entities, etc.; it should also reconstruct the users' profiles by retrieving useful information about their tastes and their preferences.

The second component, the *Recommendation Engine*, is devoted to the computation of item suggestions; it receives as input the model generated by the Model Builder and analyzes this knowledge to find semantic connections between users and items. The assumption is that, if a user profile can be connected to an item description by a set of semantic links, this means that that item is a good candidate for recommendation. Still, the component should also check for specific characteristics of those connection "paths" in order to compute a score – the so-called *utility value* – that represents the system's degree of confidence in the usefulness of such recommendation for the user.

2.2 Application Scenario: Recommending Web Services

SOA4All[2] is a European research project aimed to provide a comprehensive framework that integrates SOA, context management, Web principles, Web 2.0 and semantic technologies into a service delivery platform. SOA4All tools can be used by developers to discover, compose and reuse services in order to build a new application. In this scenario, the users of the system are developers looking

[2] Cf. `http://www.soa4all.eu/`.

for services, which are the items to be recommended. For example, a developer who wants to build an e-commerce website is interested in finding pre-existing services to be reused, like a shopping cart manager or a payment service.

Within the SOA4All project, we applied the approach sketched above to realize a Semantic Web-enabled Recommender System[3] to suggest developers (users) services of their interest (recommended items). Developers are identified by their OpenID, which is an unambiguous Web identifier; by using this identifier, the Model Builder retrieves a user description from the Web of Data which complements the "private" data about the user. Moreover, the services to be recommended have their own URIs to identify them and are semantically annotated with a categorization ontology; this lets the Model Builder find related information. Finally, the Recommendation Engine processes the SOA4All users' profiles and the services' description to find if and how they are semantically connected; if such connections exist, the engine computes their utility values in order to detect which services can be interesting for the system users.

It is worth noting that, since our Recommender System is a Semantic Web application that accesses distributed linked data sources, we developed it on top of LarKC[4] [11], an integrated and pluggable platform for Web-scale semantic computing. We configured a set of LarKC "workflows" to search for relevant sources on the (Semantic) Web and to interact with them to retrieve the desired data. A LarKC workflow is activated when it receives a request under the form of SPARQL query.

3 Building a Knowledge Base with the Web of Data

The Model Builder creates a model containing users' profiles and items' descriptions by processing the available data sources, both private and public data. This component can be further divided in three parts, as depicted in Figure 2: the User Profiler, the Item Profiler and the Linker.

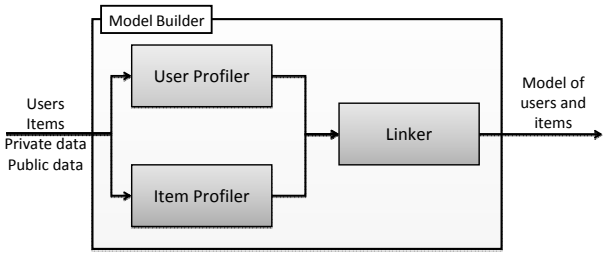

Fig. 2. The Model Builder and its components

The *User Profiler* and the *Item Profiler* respectively build a description of the users and a description of the items; the main task of the Linker is to connect

[3] Cf. `http://etechdemo.cefriel.it/rs-answers-service/`.
[4] Cf. `http://www.larkc.eu/`.

the users to the items, adding the missing data to complete the "paths" between users, their tastes and interests on the one hand, and items and their features on the other hand.

3.1 Generic Entity Profiler

Since both User and Item Profilers aim at building a semantic description of a user/item respectively, to better clarify their role, we firstly describe a generic *Entity Profiler*. Then, in Sections 3.2 and 3.3, we detail the specificities of those two components within the Model Builder.

An Entity Profiler can be characterized by the entities to be described and by some specification of the desired description of those entities, e.g. in terms of the requested entity attributes. To reconstruct the entity description, the Entity Profiler accesses both the private sources and the public data published on the Web of Data; the Entity Profiler queries and traverses those knowledge sources and retrieves the needed data; in case, it interlinks the gathered data to reconstruct a complete model of the entities to be described.

We assume that the entities to profile can be identified by URIs; this kind of identifier can be used to retrieve more information from the Web. We also consider that the entity profile will be expressed in terms of RDF triples and RDF links to external sources. In order to build such a profile, the Entity Profiler can be configured to specify what kind of profile data should be fetched; the basic assumption is a SPARQL query in the form `DESCRIBE <entity-URI>`, but configuring this component could mean defining a detailed `CONSTRUCT` query. In the latter case, the `CONTRUCT` clause could be used to transform the original retrieved data into a common format that is more suitable for the subsequent elaborations. When the entity description is fetched from several heterogeneous sources, this transformation is needed to homogenize the different pieces of data.

For what regards the data sources, the Entity Profiler should first search in the application private data; then it should look for additional data from external sources. Those public sources can either be manually identified or dynamically discovered; in the former case, the Entity Profiler can be configured with a list of known data sources (to be used, for example, in the `FROM` clause of SPARQL queries), while, in the latter case, the component should query a Semantic Web search engine service (like Sindice[5] [22] or Watson[6] [10]) and then interact with the returned list of sources to retrieve the additional information.

A final note about the "storage" of entity profiles: the most natural approach would be to store all the reconstructed profiles locally; this is also the most common approach in existing recommender systems. Growing the amount of data gathered from public sources, however, this could be unfeasible. Being the Web of Data distributed by nature, moving from a dataset to a connected one is always possible by following RDF links. A Semantic Web-enabled Recommender System could therefore have a local knowledge base for the RDF links to external

[5] Sindice `http://sindice.com/`.
[6] Watson `http://watson.kmi.open.ac.uk/`.

public datasets, leaving in their original locations a large part of the public data. Incorporating only the links to the remote locations lets the system keep its knowledge base small and manageable, without hindering the possibility to access the public data on the Web. Of course, a pragmatic solution could include partial local caching of remote data.

3.2 User Profiler

The *User Profiler* is the instantiation of the generic Entity Profiler with the goal of reconstructing the user profile. Usually, recommender systems adopt two different strategies to derive a user profile: by analyzing implicit user feedbacks and by processing explicit user inputs.

The *implicit feedbacks analysis* [14] allows to define users' needs by collecting data about their behavior, for example capturing navigation links, bookmarks and so on. The resulting user profile in this case can be inaccurate, but the advantage is that it is reconstructed without bothering users, e.g. when they cannot or do not want to provide personal details. On the other hand, when users insert specific information about themselves and their preferences, they build an *explicit profile* describing their interests. As a consequence, the system gets a very accurate user profile based on what users voluntarily provide.

Our concept of Semantic Web-enabled Recommender System does not require the adoption of either specific technique. The User Profiler addresses the challenge of *complementing* a user description, derived via implicit/explicit techniques, with additional public information from the Web of Data. Therefore, the following considerations are valid in both scenarios.

The Web of Data allows for a relevant *enrichment* of the profiles, through the discovery of different kinds of information. Not only the User Profiler can look for specific information about users and their tastes; it can also enrich the description of users' needs, by retrieving more detailed descriptions of interest topics. The availability of such information is due to the growing success of social networks; Web users are more and more accustomed to describe their profile on a multitude of different platforms and the Semantic Web community has investigated how to automatically derive a structured user profile from the social Web (e.g. extracting users' interests from their Facebook pages [27]).

In particular, we refer to *FOAF*[7] [7] profile information which, in the LOD Cloud, represents one of the largest datasets. To recommender systems, FOAF can offer useful kinds of information, like relations between users (`foaf:knows` links between profiles) and users' topics of interest (`foaf:interest` property values). Retrieving the FOAF profiles of the recommender system's users can be a successful way to obtain additional data to enrich users' profiles; for example, turning to FOAF data can be very useful when the system has no available information about new users (the so-called user cold start problem). Unfortunately, existing FOAF profiles do not always contain useful information, but we believe that, with the growing adoption of this vocabulary, more and more relevant information about people will be made available on the Web of Data.

[7] Cf. `http://www.foaf-project.org`.

Reconstructing SOA4All developers' profiles. In the scenario of service recommendation introduced in Section 2.2, system users are developers identified by their OpenID. In this context, therefore, we used this identifier to find an available FOAF profile, since FOAF specification defines the inverse functional property `foaf:openid`. If such a profile is found, we retrieve `foaf:interest`s, thus collecting the useful hints to understand users' preferences.

Since we want to build a profile that lets the recommender system provide useful service suggestions, we do not limit the developers' profiling to their FOAF description. Starting from the values of the `foaf:interest` property, we try to identify related DBpedia resources [5]. In this way, as explained later, we pave the way for an easier discovery of semantic connections between developers and the services to be recommended. An example of FOAF profile expressed in N3 retrieved from the Web of Data starting from an OpenID could be the following:

```
<user-uri> foaf:openid   <user-openid> ;
           foaf:interest dbp:ClassicalMusic .
```

Finally, the retrieved profile data are converted into a common format expressed by the Weighted Interest Ontology[8]; this data model lets us keep track of the different interest degree of a user with regards to a topic. Moreover, this ontology could be used to tell apart the different "contexts" of a user's profile: a person could be interested in books in his private life and in music for professional reasons. The example above should therefore be converted as follows:

```
<user-uri> foaf:openid   <user-openid> ;
           wi:preference [
                 rdf:type  wi:WeightedInterest ;
                 wi:topic  dbp:ClassicalMusic ;
                 wi:weight "1" ;
                 wi:context "professional life"
           ] .
```

3.3 Item Profiler

As the User Profiler builds users' description, the *Item Profiler* performs similar actions on the recommendable items. The goal of this component is to reconstruct a description of the items, collecting their relevant features and properties.

Since an item is any entity that could be recommended to the user, the specific kind of information to be retrieved strongly depends on the concrete scenario and cannot be generalized. Nonetheless, in the following we offer some hints on how Semantic Web technologies can help this component. In particular, we focus on enriching the "private data" about items with additional details from the Web of Data; as with users' profiles, we assume each item can be identified by a URI to let the Item Profiler search for related information.

For what regards the data sources to be queried, it is of course hard to indicate a source that can prove useful in any possible case. As explained in

[8] Weighted Interest Ontology `http://xmlns.notu.be/wi/`.

Section 3.1, the location of data sources of interest can be reached through the use of a Semantic Web search engine, by looking for information directly related to the item or relevant with respect to some domain-specific concept or entity.

It can also be advisable, however, to retrieve information from the most common generic sources, like the already cited DBpedia – "the crystallization point for the Web of Data" [5] – or Freebase[9]. The connection of an item description to those broad and common sources, in turn, can result in an easier discovery of semantic connections with users' profiles, as illustrated in Section 3.4.

Reconstructing SOA4All services' descriptions. In the SOA4All scenario, the recommendable items are services. Each service – either SOAP or REST – is identified by a URI and semantically described; the basic service data can be retrieved through the iServe[10] linked data endpoint developed in SOA4All. In particular, services are annotated with regards to the Service-Finder Category ontology[11], which describe general purpose classifications of services. An example of service profile retrieved in such a way is the following:

```
<service-uri> rdf:type sf:Service ;
              sawsdl:modelReference sfcat:Music .
```

As illustrated for user profiling, service description could be turned in a common format suitable for recommendation computing. A possible expressive model is offered by the Service-Finder ontology[12] as follows:

```
<service-uri> rdf:type sf:Service ;
              sf:hasCategoryAnnotation  [
                       sf:hasCategory sfcat:Music ;
                       sf:strength    "1"
              ] .
```

3.4 Linker

The output of the two profilers consists of two RDF graphs (Figure 3), one containing users' profiles and one describing recommendable items and their features. As described in Section 4, computing recommendations is the process of analyzing the relations between users and items, thus, the goal of the *Linker* is to supply the additional RDF triples describing the relations between users (or their interests) and items (or their features). Some of these connections could already be part of items/users profiles; otherwise, the Linker should look for the missing links. In [9], those three models to be connected are named *User Knowledge*, *Catalog Knowledge* (data about items' features) and *Functional Knowledge* (data about the mapping between user interests and items).

To discover useful connections, the Linker could perform two operations on the Web of Data: inference and RDF paths search. Applying *inference* on User

[9] Freebase `http://www.freebase.com/`.
[10] iServe `http://iserve.kmi.open.ac.uk/`.
[11] Cf. `http://www.service-finder.eu/ontologies/ServiceCategories`.
[12] Cf. `http://www.service-finder.eu/ontologies/ServiceOntology`.

and Catalog Knowledge, the Linker could find hidden relations among the described entities. In addition, inference could improve the RDF path search by adding new "starting points" to the data discovery.

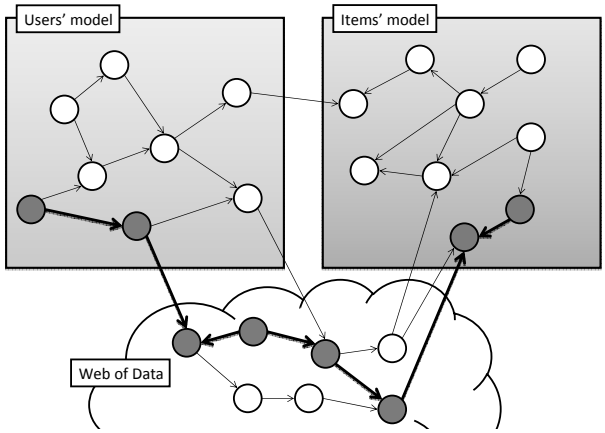

Fig. 3. Linker connection discovery

The *RDF path search* consists in the discovery of the missing links between users' and items' profiles on the Web of data. This means that, if additional connections exist, they could as well be published on the Web of Data. In literature, some works to perform RDF path search on the LOD Cloud already exist.

RelFinder[13] [16, 12] is an application that receives as input two entities and looks for paths among them. It constructs a set of SPARQL queries that ask for a path between the two entities with a predefined orientation and length; then it submits the queries to a SPARQL endpoint to retrieve the paths. RelFinder works if both the entities are accessible through the same endpoint.

SCARLET [28] copes with the problem of looking for connections between two concepts. First it looks for an ontology that contains both concepts: if it exists, the ontology is processed to derive the relation between the concepts (e.g., disjointness or equivalence). If it doesn't, SCARLET recursively looks for a set of ontologies that allows to link the two inputs.

The idea behind SCARLET could be extended from concepts to generic entities. The Linker could act in a similar way: given the two resources for which a connection should be found, it should firstly identify the datasets potentially containing the RDF paths; then, traversing the datasets by following a specific policy (for example using only a set of predefined properties), it should try to find the connections. It is worth noting that an exhaustive search for paths is not an issue, since the Linker's aim is to find useful paths for the computation of the recommendations.

Linking services to SOA4All users. As explained above, services are annotated by the use of the Service-Finder Category ontology. Those categories are

[13] RelFinder `http://relfinder.dbpedia.org/`.

mapped to DBpedia categories[14], by the use of SKOS [20] mapping properties (specifically, `skos:closeMatch`). This existing mapping enables the connection between any service annotated with the Service-Finder Category ontology with the Web of Data. Our Linker, therefore, starting from the service categorization recalls the link between such categorization and DBpedia categories from the Service-Finder mapping, fetching additional knowledge, as follows:

```
sfcat:Music skos:closeMatch dbpcat:Music .
```

Moreover, DBpedia categories constitute a taxonomy, in which categories are related via the `skos:broader` mapping property. DBpedia topics are then related to DBpedia categories by a `skos:subject` predicate. Thus, the Linker can find a path between user interests (expressed as DBpedia topics) and service categorizations (mapped to DBpedia categories). The following triples complete the path between the user and the item of the previous listings:

```
dbp:ClassicalMusic    skos:subject  dbpcat:ClassicalMusic .
dbpcat:ClassicalMusic skos:broader  dbpcat:MusicGenre .
dbpcat:MusicGenre     skos:broader  dbpcat:Music .
```

4 Computing Recommendations with the Semantic Web

The last element in our vision for a Semantic Web Recommender System is of course the component to generate suggestions of interesting items to the users: the *Recommendation Engine*. On the basis of the RDF graph that connects users to items as computed by the Model Builder, this component must derive a list of recommendations, i.e. a list of user-item pairs qualified by a *utility value* which represents the confidence in the predicted user/item correlation.

The approach of our Recommendation Engine is oriented to find *meaningful relations* between users and items in the descriptive graph. The first step is thus identifying the relevant path(s) – in terms of RDF triples – that connects a user with an item within the graph reconstructed by the Linker. It is possible that no path exists to connect a user to an item: in this case, the Engine does not produce any recommendation for that pair; on the other hand, when multiple paths exist, all paths or only a subset of them can be considered. In any case, each path must be evaluated and given a score.

In order to judge if an item can be of interest for a user, the bare existence of a path between them is not enough; indeed, the path should contain evidence of the recommendation "utility". To this end, Semantic Web technologies can play again an important role: the RDF path in fact is not only a route connecting two points in a graph, but it consists in a "semantic" description of the reasons why the user and the item are linked.

Under this perspective, the Recommendation Engine must check the "content" of the user-item connection path, to identify signs of potential utility. For

[14] Cf. `http://www.service-finder.eu/ontologies/SFC2DBpedia`.

example, it could verify if the path contains triples that express interest, liking or importance (e.g. the user *likes* a topic which is related to the item); on the contrary, it should make sure that the path does not contain expressions of disapproval or distaste (e.g. the user *dislikes* a subject to which the item refers). Finally, it could take into account the user "context" or "role" to give higher scores to paths relevant for the current user needs. Therefore, the Recommendation Engine should *verify a set of constraints* within the RDF path(s); the satisfaction of each constraint leads to the attribution of a score and the combination of all scores constitutes the utility value for the user-item pair.

Pragmatically, those constraints to be verified can be expressed in a number of different ways using Semantic Web technologies, including SWRL [13] or RIF [6] rules. In the simplest case, those constraints can be expressed as *triple patterns* to be verified via a *SPARQL query* [29]: the query verifies specific path characteristics, like triples with specific predicates. To configure a Recommender Engine, a set of constraints should be provided, each one qualified by a score that, in case of verification, contributes to the utility value computation. Once all those computations take place, making item recommendations for a specific user means selecting the user-item pairs with the highest utility values.

Moreover, when suggesting those recommendations to the user, the Recommendation Engine could also provide a *proof* for its choice, i.e. the path connecting the user to the recommended item. This proof lets the user understand the "semantics" of the recommendation and check its validity; in turn, this could enable the possibility to gather feedbacks from the user about the soundness of a recommendation, thus paving the way for an improvement of the user/item model (e.g. the user could add/modify his interests in order to get more tailored suggestions).

Computing service recommendations scores. In our service recommendation scenario, we compute the utility value by analyzing the paths. On the one hand, we rely on the knowledge about users and items: user interests are "weighted" as exemplified before and service annotations have a "strength" representing how much the employed automatic annotator software is confident about the categorization. On the other hand, the RDF path connecting a user to a service contains other information, like the DBpedia topics and categories.

For each user-service couple, the system computes a correlation value out of all the paths connecting them. Given a path, a first utility value is computed considering specific characteristics of the RDF path, like the number of `skos:broader` relations between DBpedia categories (the fewer relations, the higher the score) and the presence of DBpedia "top" categories (if present, the connection path is probably not very meaningful). Then, this value is combined with the interest's weight and the categorization's strength, in order to compute a global utility value for the path.

5 Related works

Semantic Web and Recommender Systems are two research fields with several points of contact: in literature, it is possible to find several works related to the study of their interaction [26].

A first way to use Semantic Web technologies in recommender systems is to *describe users' profiles and items' features*. Middleton, Alani and De Roure in [19] use ontologies to model the topics taxonomy of research papers (the items to be recommended) and users' interests, in order to compute recommendations and to identify meaningful groups of users (communities of practice). In [2], Bannwart et al. present OMORE, a Content-based Recommender System for movies; it runs as a Firefox plug-in and it predicts how much a movie could be of interest for a user. When running, the application analyzes the Web pages the user views; if a page is related to a movie, OMORE processes it, retrieving the features of the movie through the LOD Cloud and, thus, profiling the user.

Regarding the use of Semantic Web tools for the *generation of recommendations*, Abel et al. in [1] present a recommender system for an on-line community with the goal of suggesting relevant discussions to the users. To process the recommendations, the system can select among several collaborative filtering algorithms; those algorithms are exposed as Semantic Web Services described using OWL [18]. To choose the best algorithm, the system employs a Semantic Web rule-based engine (with rules defined in SWRL [13]).

In [17], Manikrao and Prabhakar present a recommender system for Web services. One of the core components of their system is a semantic matcher, that operates on a knowledge-base in order to match users' needs with services. Knowledge-based Recommender System can take advantage of Semantic Web tools like reasoners and inference engines; to our best knowledge, however, the use of those technologies in recommender systems is still quite limited.

Another way Semantic Web could help to build a Knowledge-based Recommender System is *to build recommendations' proof*. Passant and Decker in [23] present dbrec, a recommender system to compute music recommendations. The system gets description of musicians and music bands from DBpedia and processes the retrieved RDF graph in order to compute a semantic similarity value (called Linked Data Semantic Distance) among artists. When users navigate the dbrec Web site, the system shows the reasons behind the recommendations of similar musicians.

Finally, Semantic Web technologies can be successfully employed to *integrate data from heterogeneous sources*. Passant and Raimond in [24] propose the use of the data in the LOD Cloud to build a music recommender system. They analyze three different kinds of data available on the LOD Cloud that could be used to compute recommendations: social-network data (the FOAF-o-sphere), descriptions of artists and bands and user-defined tags. Those data are distributed on several sources, such as DBpedia or MusicBrainz, and they are integrated and interlinked by following the Linked Data principles [3].

Another example of Semantic Web-based integration can be found in [31], where Szomszor et al. cope with the problem of building a unified knowledge-

base to compute recommendations from two datasets (IMDb and Netflix). To achieve their goal, they employ RDF and make the integrated data accessible via a SPARQL endpoint.

6 Conclusions

In this paper we presented our vision of how a Knowledge based Recommender System could be based on and exploit all the capabilities of Semantic Web technologies. We explained how the model of users and items can be enriched with the knowledge from the Web of Data and how Semantic Web tools can be employed to elaborate data and support generating recommendations. We designed the structure and the main modules that compose such a system at a conceptual level, giving hints on how this can be realized in practice; additionally, we illustrated how we concretely built a demonstrator of such a Semantic Web-enabled Recommender System in the context of service recommendations in the SOA4All project.

Our main goal with this paper was to present several opportunities to leverage the results of the Semantic Web community in the Recommender Systems field; we hope that other researchers can take our concept and analysis as reference for further investigations on the possible interplay of the two "worlds".

The pure adoption of Semantic Web technologies, however, does not solve some existing open issues in building a Knowledge-based Recommender System. In particular, since the quality of recommendations is directly correlated with the *quality of data*, relying on the knowledge in the LOD Cloud means trusting the quality of its contents. To dispel the concerns about the quality of data and the level of "trust" of the linked data sources on the Web, the Semantic Web community is putting a lot of effort in the definition and standardization of "*provenance*" descriptions[15] [21]. With the arrival of massive amounts of Semantic Web data, information about the origin of that data becomes an important factor in telling apart reliable data from low-quality information.

In an open and inclusive environment such as the Web, information is often contradictory or questionable; the distributed and redundant nature of the Web puts at risk not only the *consistency of knowledge*, but also its *completeness and reliability*. To overcome those problems, usually people make trust judgments based on what they know about the data provenance. However, when provenance information is not available or is not enough, the problem of managing inconsistency remains open. This, in turn, has a direct consequence on the possibility to evaluate the generated recommendations, e.g. in terms of *precision* and *recall*. On the other hand, the employment of semantic descriptions paves the way to a recommendation evaluation based on the *meaning* of data.

All in all, our proposed approach is far from being complete. Our future work will be devoted to extend and improve the current proof of concept, to evaluate it and to detail the lessons learned from the adoption of our approach. In our roadmap, we would like to investigate in the direction of deriving user profiles from implicit knowledge (as in our previous work [32]) and of leveraging the

[15] Cf. `http://www.w3.org/2005/Incubator/prov/`.

semantic relations between users and between items. Finally we would like to explore the feasibility of realizing a hybrid system [9] by joining our Semantic Web-enabled Recommender System with a Collaborative Filtering approach.

Acknowledgments

This research has been partially supported by the *SOA4All* (FP7-IST-215219), *LarKC* (FP7-IST-215535) and *Service-Finder* (FP7-IST-215876) EU co-funded projects.

References

1. F. Abel, I. I. Bittencourt, N. Henze, D. Krause, and J. Vassileva. A Rule-Based Recommender System for Online Discussion Forums. In *Proceedings of the 5th international conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH '08)*, pages 12–21, Berlin, Heidelberg, 2008. Springer-Verlag.
2. T. Bannwart, A. Bouza, G. Reif, and A. Bernstein. Private Cross-page Movie Recommendations with the Firefox add-on OMORE. In *8th International Semantic Web Conference (ISWC2009)*, Washington DC, USA, October 2009.
3. T. Berners-Lee. *Linked Data – Design Issues.* Online at http://www.w3.org/DesignIssues/LinkedData.html, 2006.
4. C. Bizer, T. Heath, and T. Berners-Lee. Linked Data – The Story So Far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.
5. C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. DBpedia – A Crystallization Point for the Web of Data. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 7:154–165, 2009.
6. H. Boley, G. Hallmark, M. Kifer, A. Paschke, A. Polleres, and D. Reynold. *RIF Core Dialect – W3C Recommendation.* Available at http://www.w3.org/TR/rif-core/, June 22th, 2010.
7. D. Brickley and L. Miller. *FOAF Vocabulary Specification 0.97.* Available at http://xmlns.com/foaf/spec/, January 1st, 2010.
8. R. Burke. Knowledge-Based Recommender Systems. *Encyclopedia of Library and Information Science*, 69(32), 2000.
9. R. Burke. Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.
10. M. d'Aquin, C. Baldassarre, L. Gridinoc, S. Angeletou, M. Sabou, and E. Motta. Characterizing Knowledge on the Semantic Web with Watson. In *Proceedings of the 5th International Workshop on Evaluation of Ontologies and Ontology-based Tools (EON2007), co-located with the ISWC2007*, pages 1–10, Busan, Korea, 2007.
11. D. Fensel, F. van Harmelen, B. Andersson, P. Brennan, H. Cunningham, E. Della Valle, F. Fischer, Z. Huang, A. Kiryakov, T. Kyung-il Lee, L. School, V. Tresp, S. Wesner, M. Witbrock, and N. Zhong. Towards LarKC: a Platform for Web-scale Reasoning, 8 2008.
12. P. Heim, S. Hellmann, J. Lehmann, S. Lohmann, and T. Stegemann. RelFinder: Revealing Relationships in RDF Knowledge Bases. In *Proceedings of the 4th International Conference on Semantic and Digital Media Technologies (SAMT 2009)*, volume 5887 of *Lecture Notes in Computer Science*, pages 182–187. Springer, 2009.
13. I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. *SWRL: A Semantic Web Rule Language Combining OWL and RuleML.* Available at http://www.w3.org/Submission/SWRL/, 2004.
14. D. Kelly and J. Teevan. Implicit feedback for inferring user preference: a bibliography. *SIGIR Forum*, 37(2):18–28, 2003.

15. G. Klyne and J. J. Carroll. *Resource Description Framework (RDF): Concepts and Abstract Syntax – W3C Recommendation.* Available at http://www.w3.org/TR/rdf-concepts/, 2004.
16. J. Lehmann, J. Schüppel, and S. Auer. Discovering Unknown Connections - the DBpedia Relationship Finder. In *Proceedings of the 1st SABRE Conference on Social Semantic Web*, 2007.
17. U. S. Manikrao and T. V. Prabhakar. Dynamic Selection of Web Services with Recommendation System. In *Proceedings of the International Conference on Next Generation Web Services Practices (NWESP '05)*, page 117, Washington, DC, USA, 2005. IEEE Computer Society.
18. D. L. McGuinness and F. van Harmelen. *OWL Web Ontology Language Overview – W3C Recommendation.* Available at http://www.w3.org/TR/owl-features/, 2004.
19. S. E. Middleton, H. Alani, and D. C. De Roure. Exploiting Synergy Between Ontologies and Recommender Systems. In *Proceedings of the WWW2002 International Workshop on the Semantic Web*, 2002.
20. A. Miles and S. Bechhofer. *SKOS Simple Knowledge Organization System Reference – W3C Recommendation.* Available at http://www.w3.org/TR/skos-reference/, August 18th, 2009.
21. L. Moreau. The foundations for provenance on the web. *Foundations and Trends in Web Science*, November 2009.
22. E. Oren, R. Delbru, M. Catasta, R. Cyganiak, H. Stenzhorn, and G. Tummarello. Sindice.com: a document-oriented lookup index for open linked data. *International Journal of Metadata, Semantics and Ontologies*, 3(1):37–52, 2008.
23. A. Passant and S. Decker. Hey! Ho! Let's Go! Explanatory Music Recommendations with dbrec. In *ESWC Part II*, pages 411–415, 2010.
24. A. Passant and Y. Raimond. Combining Social Music and Semantic Web for Music-Related Recommender Systems. In *Proceedings of the 1st Workshop on Social Data on the Web (SDoW2008), co-located with the ISWC2008*, Karlsruhe, Deutschland, October 2008.
25. M. J. Pazzani and D. Billsus. Content-Based Recommendation Systems. In *The Adaptive Web*, pages 325–341, 2007.
26. E. Peis, J. M. M. del Castillo, and J. A. Delgado-López. Semantic Recommender Systems. Analysis of the state of the topic. *Hipertext.net*, 6:online, 2008.
27. M. Rowe and F. Ciravegna. Getting to Me – Exporting Semantic Social Network from Facebook. In *Proceedings of the 1st Workshop on Social Data on the Web (SDoW2008), co-located with the ISWC2008*, 2008.
28. M. Sabou, M. d'Aquin, and E. Motta. SCARLET: SemantiC relAtion discoveRy by harvesting onLinE onTologies. In *Proceedings of the 5th European Semantic Web Conference (ESWC2008)*, Tenerife, Spain, 2008.
29. A. Seaborne and E. Prud'hommeaux. *SPARQL Query Language for RDF – W3C Recommendation.* Available at http://www.w3.org/TR/rdf-sparql-query/, January 15th, 2008.
30. X. Su and T. M. Khoshgoftaar. A Survey of Collaborative Filtering Techniques. *Advances in Artificial Intelligence*, vol. 2009(Article ID 421425), 2009.
31. M. Szomszor, C. Cattuto, H. Alani, K. O'Hara, A. Baldassarri, V. Loreto, and V. D. Servedio. Folksonomies, the Semantic Web, and Movie Recommendation . In *Bridging the Gap between Semantic Web and Web 2.0 workshop, colocated with ESWC2007*, 2007.
32. A. Turati, D. Cerizza, I. Celino, and E. Della Valle. Analyzing User Actions within a Web 2.0 Portal to Improve a Collaborative Filtering Recommendation System. In *Web Intelligence/IAT Workshops*, pages 65–68. IEEE, 2009.