Linked Open Data to support Content-based Recommender Systems

Tommaso Di Noia[1], Roberto Mirizzi[1],
Vito Claudio Ostuni[1], Davide Romito[1], Markus Zanker[2*]
[1]Politecnico di Bari – Via Orabona, 4 – 70125 Bari, Italy
[2]Inst. of Applied Informatics, University Klagenfurt – Universitätstraße, 65-67 – 9020 Klagenfurt, Austria
t.dinoia@poliba.it, {mirizzi,ostuni,d.romito}@deemail.poliba.it,
markus.zanker@uni-klu.ac.at

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 spread and to the more recent `Linked Open Data` (`LOD`) initiative, a vast amount of `RDF` data have been published in freely accessible datasets. These datasets are connected with each other to form the so called `Linked Open Data` cloud. As of today, there are tons of `RDF` data available in the Web of Data, but only few applications really exploit their potential power. In this paper we show how these data can successfully be used to develop a recommender system (RS) that relies exclusively on the information encoded in the Web of Data. We implemented a content-based RS that leverages the data available within `Linked Open Data` datasets (in particular `DBpedia`, `Freebase` and `LinkedMDB`) in order to recommend movies to the end users. We extensively evaluated the approach and validated the effectiveness of the algorithms by experimentally measuring their accuracy with precision and recall metrics.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval

General Terms

Algorithms, Experimentation, Performance

Keywords

Content-based Recommender Systems, Vector Space Model, Linked Data, DBpedia, LinkedMDB, Freebase, Semantic Web, MovieLens, Precision, Recall

*The authors are listed in alphabetical order.

I-SEMANTICS 2012, 8th Int. Conf. on Semantic Systems, Sept. 5-7, 2012, Graz, Austria.

1. INTRODUCTION

With the pervasive spread of published data in the Web 2.0, we entered into an era of *information overload*: more information is produced than we can really consume and process. In order to cope with such a problem, there has been a growing number of systems that try to support their users when searching for information. Among them, we mention a specific family of such systems for information filtering: **recommender systems** [20], whose objective is supporting users to make choices. Currently, state of the art applications for content-based recommendation mainly (but not exclusively) exploit the information coming from different text sources in order to identify keywords, or patterns of keywords able to model both the user profile and the resources to be suggested. In this paper we show how datasets from the `Linked Open Data` cloud [4] are mature enough and rich in high quality data to be used as the main information source for a complete application that offers the functionalities of a semantic content-based recommender system. Indeed, although recommender systems leverage well established technologies and tools, new challenges arise when they exploit the huge amount of interlinked data coming from the Semantic Web.

The main contributions of this paper may be summarized as follows:

- *Creation of a content-based recommender system exploiting exclusively `LOD` datasets.* Our RS is fed by data coming from `DBpedia` [5], `LinkedMDB` [12] and `Freebase` [6]. We propose a vector space model approach to compute similarities between `RDF` resources in `LOD`. In line with the trends on the social Web 2.0, we allow a binary user rating (i.e., *like – don't like*). We propose and compare several formulas for recommendation and different approaches for training the system.
- *Evaluation.* Our system has been extensively evaluated both by performance analysis and by standard precision and recall measures against the `MovieLens` [14] dataset. Although the general approach we present here is domain independent, we selected the movie domain in order to ease the comparison with other non-`LOD` systems.

The remainder of the paper is structured as follows: in Section 2 we detail how we leverage information contained in `Linked Open Data` to compute semantic similarities between resources, then in Section 3 we describe our recommendation algorithm. In Section 4 we evaluate the proposed approach, while in Section 5 we review relevant related work. Conclusion and future work close the paper.

2. COMPUTING SIMILARITY IN LOD DATASETS

Content-based (CB) recommender systems suggest new items to the user based on a similarity value computed between the description of items previously selected by the user and the description of new unknown items. Several approaches have been proposed in the literature to evaluate such similarity functions when dealing with keyword-based information where both the items and the query are usually represented as weighted vectors of terms. In such a setting, one of the most popular approaches is the one combining the *term-frequency/inverse-frequency* and the *cosine similarity* measure. The former is used to assign a score (weight) to the keywords occurring in the vectors, while the latter evaluates the similarity between two items or between the query and an item [1].

In CB recommender systems, the module in charge of extracting relevant information from items description and of representing it as a vector of keywords is the so called *Content Analyzer* (CA) [15]. It mainly uses some Natural Language Processing techniques to extract/disambiguate/expand keywords in order to create a model of the item description. The use of `Linked Open Data` datasets to retrieve information related to an item eases the pre-processing steps performed by the CA since the information is already structured in an ontological way. Moreover, depending on the dataset, there is the availability of data related to diverse knowledge domains. If we consider datasets such as `DBpedia` or `Freebase`, we are able to access to rich linked data referring to a high variety of topics.

In the following, for the sake of presentation we will refer to `LOD` data related to the movie domain in three different datasets, i.e., `DBpedia`, `Freebase` and `LinkedMDB`. Nevertheless, it is worth noticing that the overall approach is domain and dataset independent and can be adapted to other knowledge domains.

Thanks to their `SPARQL` endpoints, we can quite easily extract portions related to the movie domain from `Linked Open Data` datasets. We use this information as the base for our recommender system. Intuitively, we assume that if two movies share some information (e.g., part of the cast, the director, the genre, some categories, etc.), then they are related with each other. Roughly speaking, the more features two movies have in common, the more they are similar. In a few words, if we consider for instance `DBpedia`, a similarity between two movies (or two resources in general) can be detected if in the `RDF` graph:

- they are directly related: this happens for example if a movie is the sequel of another movie. In `DBpedia` this state is handled by the properties `dbpedia-owl:subsequentWork` and `dbpedia-owl:previousWork`.
- they are the subject of two `RDF` triples having the same property and the same object, as for example when two movies have the same director. In the movie domain, we take into account about 20 properties, such as `dbpedia-owl:starring` and `dbpedia-owl:director`. They have been automatically extracted via `SPARQL` queries (see Section 4.1). In particular, the property `dcterms:subject` relates a resource to its categories. Categories are used in `Wikipedia` to organize the entire project, and to help users to give a structure to the knowledge base, by grouping together pages on the same subject. In `DBpedia`, the hierarchical structure of the categories is modeled trough the property `skos:broader`. We consider also this property in our approach. This allows us to catch implicit relations and hidden information, i.e., information that is not directly detectable just looking at the nearest neighbors in the `RDF` graph (cf. the highlighted path in Figure 1).
- they are the object of two `RDF` triples having the same property and the same subject.

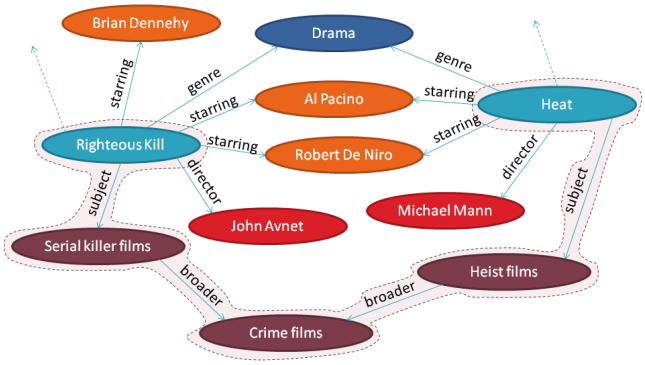


Figure 1: A sample of an `RDF` graph related to the movie domain.

Figure 1 shows a sample of the `RDF` graph containing properties and resources coming both from `DBpedia` and from `LinkedMDB/IMDB`.

2.1 VSM for LOD

In order to compute the similarities between movies, we adapt to a `LOD`-based setting one of the most popular models in classic information retrieval: the Vector Space Model (VSM) [21]. We want to stress here that, among the various algorithms to compute similarities, such as *Support Vector Machines* and other classifiers [2], we selected the VSM in order to have a good baseline for further developments and comparisons. In VSM non-binary weights are assigned to index terms in queries and in documents (represented as sets of terms), and are used to compute the degree of similarity between each document in the collection and the query. In our approach, we semanticized the classical VSM, usually used for text retrieval, to deal with `RDF` graphs. In a nutshell, we represent the whole `RDF` graph as a 3-dimensional matrix where each slice refers to an ontology property and represents its adjacency matrix. A component (i.e. a cell in the matrix) is not *null* if there is a property that relates a subject (on the rows) to an object (on the columns). Given a property, each movie is seen as a vector, whose components refer to the *term frequency-inverse document frequency* TF-IDF (or better, in this case, *resource frequency-inverse movie frequency*). For a given slice (i.e. a particular property), the similarity degree between two movies is the correlation between the two vectors, and it is quantified by the cosine of the angle between them. All the nodes of the graph are represented both on the rows and on the columns of the matrix. A few words need to be spent for the properties `dcterms:subject` and `skos:broader` which are very popular in the `DBpedia` dataset. As also shown in Figure 1, every movie is related to a category by the property `dcterms:subject`

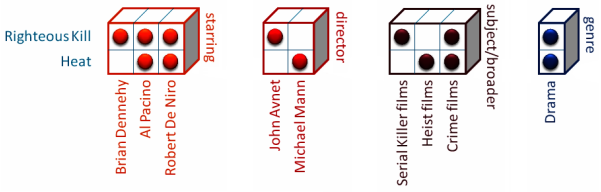


Figure 2: Matrix representation of the RDF graph of Figure 1.

which is in turn related to other categories via `skos:broader` organized in a hierarchical structure. To the purpose of the recommendation, we consider `skos:broader` as *one-step transitive*. We will explain this notion with the aid of a simple example. Suppose to have the following RDF statements:

```
dbpedia:Righteous_Kill
    dcterms:subject dbpedia:Category:Serial_killer_films .
dbpedia:Category:Serial_killer_films
    skos:broader dbpedia:Category:Crime_films .
```

Starting from `dbpedia:Category:Serial_killer_films` we have that `dbpedia:Category:Crime_films` is at a distance of one step. Hence, by considering a *one-step transitivity*, we have that:

```
dbpedia:Righteous_Kill
    dcterms:subject dbpedia:Category:Crime_films .
```

is inferred by the original statements.

Looking at the model, we may observe and remember that: (1) the matrix is very sparse; (2) we consider properties as independent with each other (there is no `rdfs:subPropertyOf` relation); (3) we are interested in discovering the similarities between movies (or in general between resources of the same `rdf:type` and not between each pair of resources). Based on the above observations, we can decompose the matrix slices into smaller matrices where each matrix refers to a specific RDF property, as shown in Figure 2. In other words, for each matrix, the rows represent somehow the *domain* of the considered property, while the columns its *range*. For a given property, the components of each row represent the contribution of a resource (i.e. an actor, a director, etc.) to the corresponding movie. With respect to a selected property p, a movie m is then represented by a vector containing all the terms/nodes related to m via p. As for classical Information Retrieval, the index terms $k_{n,p}$, that is all the nodes n linked to a movie by a specific property p, are assumed to be all mutually independent and are represented as unit vectors of a t-dimensional space, where t is the total number of index terms. Referring to Figure 2, the index terms for the *starring* property are *Brian Dennehy*, *Al Pacino* and *Robert De Niro*, while $t = 3$ is the number of all the actors that are objects of a triple involving *starring*. The representation of a movie m_i, according to the property p, is a t-dimensional vector given by $\overrightarrow{m_{i,p}} = (w_{1,i,p}, w_{2,i,p}, ..., w_{t,i,p})$, where $w_{n,i,p}$ is a non-negative and non-binary value representing the weight associated with a term-movie pair $(k_{n,p}, \overrightarrow{m_{i,p}})$. The weights $w_{n,i,p}$ we adopt in our model are TF-IDF weights. More precisely, the TF ($f_{n,i,p}$) is the frequency of the node n, as the object of an RDF triple having p as property and the node i as subject (the movie). Actually, this term can be either 0 (if i is not related to n via p) or 1, since two identical triples can not coexist in an RDF graph. As for classical information retrieval, the IDF is computed as the logarithm of the ratio between M, that is the total number of movies in the collection, and $a_{n,p}$, that is the number of movies that are linked to the resource n, by means of the predicate p. As an example, referring to Figure 2, for the *starring* property, and considering $n = AlPacino$, then $a_{AlPacino,starring}$ is equal to 2, and it represents the number of movies where *Al Pacino* acted. Relying on the model presented above, each movie can be represented as a $t \times P$ matrix, where P is the total number of selected properties. If we consider a projection on a property p, each pair of movies, m_i and m_j, are represented as t-dimensional vectors. We evaluate the degree of similarity between m_i and m_j with respect to p, as the correlation between the vectors $\overrightarrow{m_{i,p}}$ and $\overrightarrow{m_{j,p}}$. More precisely we calculate the cosine of the angle between the two vectors as:

$$sim^p(m_i, m_j) = \frac{\sum_{n=1}^{t} w_{n,i,p} \cdot w_{n,j,p}}{\sqrt{\sum_{n=1}^{t} w_{n,i,p}^2} \cdot \sqrt{\sum_{n=1}^{t} w_{n,j,p}^2}}$$

Such a value is the building block of our content-based recommender system. By means of the computed similarities, it is possible to ask the system questions like *"Which are the most similar movies to movie m_i according to a specific property p?"*, and also "*Which are the most similar movies to movie m_i according to the whole knowledge base?*".

The method described so far is general enough and it can be applied when the similarity has to be found between resources that appear as subjects of RDF triples. When the resources to be ranked appear as objects of RDF triples, it is simply a matter of swapping the rows with the columns in the matrices of Figure 2 and applying again the same algorithm. Lastly, when two resources are directly related by some specific properties (as the case of the property `dbpedia:subsequentWork`), we simply operate a matrix transformation to handle this case the same way as done so far. In the following we will see how to combine such similarity values with a user profile to compute a content-based recommendation.

3. SEMANTIC CONTENT-BASED RECOMMENDER SYSTEM

If we want to provide an answer also to questions like *"Which are the most similar movies to movie m_i according to the user profile?"* we need to go a step further by representing and exploiting the user profile. In our setting, we model it based on a binary rating such as *I like*/*I don't like* (as the one adopted by YouTube). Empirical studies on real uses cases, as the one reported in the official YouTube Blog[1], show that even if users are allowed to rate an item on a five stars scale, it can happen that they actually either give the highest grade or do not give feedback at all. Therefore, we model ratings only on a binary scale. Hence, we model the profile of the user u as:

$$profile(u) = \{\langle m_j, v_j \rangle \mid v_j = 1 \text{ if } u \text{ likes } m_j, \; v_j = -1 \text{ otherwise}\} \quad (1)$$

[1] `http://youtube-global.blogspot.it/2009/09/five-stars-dominate-ratings.html`

In order to evaluate if a new resource (movie) m_i might be of interest for u – with $\langle m_i, v_i \rangle \notin profile(u)$ – we need to combine the similarity values related to each single property of m_i and compute an overall similarity value $\tilde{r}(u, m_i)$. We evaluated two different formulas trying to find the one best performing in terms of precision and recall (see Section 4 for the results):

$$\tilde{r}(u, m_i) = \frac{\sum_{m_j \in profile(u)} v_j \cdot \frac{\sum_p \alpha_p \cdot sim^p(m_j, m_i)}{P}}{|profile(u)|} \quad (2)$$

$$\tilde{r}(u, m_i) = \frac{\sum_{m_j \in profile(u)} v_j \cdot \frac{\sum_p \alpha_p \cdot sim^p(m_j, m_i)}{\sum_p \alpha_p}}{|profile(u)|} \quad (3)$$

where P represents the number of properties we selected from the datasets we consider (see Section 2.1) and $|profile(u)|$ is the cardinality of the set $profile(u)$. The formulas take into account the similarities between the corresponding properties of the new item m_i and $m_j \in profile(u)$. A weight α_p is assigned to each property representing its worth with respect to the user profile. The difference between the two formulas is the normalization factor: in Equation 2 the similarity between m_i and m_j is divided by P, while in Equation 3 it is divided by the sum of α_p coefficients.

3.1 Training the system

The system automatically computes default values for α_p by training the model via a genetic algorithm. We describe also another approach to automatic extraction of default weights, based on `Amazon`. In the evaluation section (cf. Section 4.2 and Figure 3(a)) we will compare the results obtained by these approaches.

3.1.1 Training via a Genetic Algorithm

Genetic Algorithms (GA) are adaptive algorithms that identify a population of solutions by following the paradigm of evolution and genetics [23]. Extensive research on them has proven their effectiveness in solving a large range of optimization problems, including feature selection and weighting tasks. In GAs a fitness function is applied to evaluate individuals, and the success of the reproduction of the population varies with the fitness function. In our case, the fitness function *minimizes the misclassification error* on the training data thus improving the precision results. The evolutionary process starts from a population of α_p coefficients that are generated randomly. Then, in each generation, the fitness function evaluates every possible solution and then multiple solutions are stochastically chosen from the current population, recombined and randomly mutated to obtain a new population. This population is used in the subsequent iteration. In our tests, we varied the number of iterations to decide when the algorithm had to terminate. We observed the quality of the recommendations – measured in terms of Precision and Recall on the test set – does not improve significantly if we consider a number of iterations greater than 100 and a population size greater than 300, while the computational cost increases drastically. For this reason, we set the maximum number of iterations equal to 100 and the population size equal to 300 individuals.

In Section 4 we will discuss the performances of Equation 2 and Equation 3 in terms of the well-known accuracy measures Precision and Recall.

3.1.2 Training via Amazon

The Genetic Algorithm above described computes the optimum values for the α_p coefficients, according to a specific user profile $profile(u)$. However, in case such profile is not available – this happens for example when a new user starts using the application – we use default weights computed via a statistical analysis on `Amazon`'s collaborative recommender system. More in detail, we collected a set of 1000 randomly selected movies from `Amazon` and we checked *why* users that have bought a movie m_i also bought movie m_j, analyzing the first items in the recommendation list. For example, for the movie *Righteous Kill*, the first movie suggested by `Amazon` is *Heat*. Then, looking into our semantic graph, we checked what these two movies have in common. In particular, since these movies share part of the cast, the genre and some categories (cf. Figure 1), we assign an initial value equal to 1 to $\alpha_{starring}$, α_{genre} and $\alpha_{subject/broader}$. The main idea behind this assignment is that a user likes a movie m_i, given another movie m_j, because the two movies are similar according to some properties $p_1, \ldots, p_P$. More precisely, each time there is a path that relates two movies with each other via some property p, a counter associated to p is incremented. We iterate the process on the training movie set, and then we normalize the counters to the highest one, to finally obtain the coefficients α_p in the range $[0, 1]$.

3.2 Browsing and Explanation

One of the aspects in favor of content-based recommender systems compared to those based on collaborative filtering, is its transparency [15]. Explanations can be computed by explicitly listing, for each property, the values which are common between the movies in the user profile and the suggested one. Computing an explanation for a recommended item is crucial for the acceptance of a recommender system [14]. To this aim, the interlinked and structured nature of `LOD` datasets plays an important role since it eases the computation of a human-understandable explanation because it allows the user to explore [27] the results space following different dimensions (each one corresponding to an ontological property). By adopting a faceted browser (see [3] as an example) the user may discover new information on returned movies or even new unknown movies related to the suggested ones.

4. EVALUATION

In order to assess the performance and the quality of our recommendation algorithm, we analyzed the amount of time required to compute the rankings for the whole dataset we extracted from `DBpedia`, `Freebase` and `LinkedMDB` and we conducted several precision and recall experiments to evaluate the proposed recommendations.

4.1 Dataset and performance analysis

In the current `DBpedia` release (3.7) there are 60,194 movies. 3,524,733 triples have a movie as subject, almost two-thirds of them (2,288,968) link a movie to a resource (e.g., an *actor* URI, a *director* URI, and not a literal). More precisely, there are 575 distinct properties of such type whose domain is a movie. In our analysis we considered `dcterms:subject`,

skos:broader and the properties belonging to the DBpedia Ontology[2]. In particular, we consider dcterms:subject and skos:broader very relevant for our approach since they encode most of the ontological knowledge in the DBpedia dataset. This will be more evident in Section 4.2. Moreover, in DBpedia we use only the properties belonging to the DBpedia Ontology, because they allow users to deal with high-quality, clean and well-structured data. A list of all the properties related to the *Film* class in the DBpedia ontology is available at http://mappings.dbpedia.org/server/ontology/classes/Film. They can be easily retrieved via the following SPARQL query:

```
SELECT ?p, COUNT(?p) AS ?n WHERE {
    ?s a dbpedia-owl:Film .
    ?s ?p ?o .
    FILTER(regex(?p, "^http://dbpedia.org/ontology/"))
}
GROUP BY ?p
ORDER BY DESC(?n)
```

In the movie dataset extraction, we considered all these properties, except for the properties dbpedia-owl:thumbnail and dbpedia-owl:wikiPageExternalLink, since they clearly do not give any useful semantic contribution to our computation. Starting from the set of resources in DBpedia representing movies, we extracted related information from LinkedMDB and Freebase by following the owl:sameAs property. Summing up the most relevant characteristics of the movie subgraph extracted from DBpedia, Freebase and LinkedMDB, there are 181,914 triples involving 53,840 distinct *actors*, 54,504 triples referring to 18,149 different *directors*; 68,393 triples concerning 29,352 distinct *writers*; 368,675 triples related to *categories*. We collected a total of 27,035 categories from DBpedia, 667 genres from Freebase and 26 genres from LinkedMDB.

After the movie subgraph has been extracted, we have measured the runtime performance of the ranking process executed by our algorithm. The program is written in Java and makes extensive use of multi-threading, with 150 concurrent threads. The computation time for the recommendation of the whole extracted dataset lasted 24 minutes and 13 seconds on a dedicated server machine with 4 Xeon quadcore 2.93GHz processors and 32GB RAM. Being an extension of the classical Vector Space Model, our approach has the same time complexity: it is linear in the number of documents (i.e., movies) in the collection. Moreover, several optimizations based on heuristics can be applied straight for speeding up the computation [16].

4.2 Recommender System evaluation

In order to evaluate the quality of our algorithm, we performed the evaluation on MovieLens [14], the historical dataset for movie recommender systems. The 1M dataset contains 100,000,029 anonymous ratings from 6,040 users on 3,952 movies. MovieLens datasets are mainly aimed at evaluating collaborative recommender systems in the movie domain. Since our approach is based on a content-based recommendation, in order to use such datasets to test the performances of our algorithms, we linked resources represented in MovieLens to DBpedia ones[3]. We extracted the value of rdfs:label property from all the movies in DBpedia, together with the year of production, by using the following SPARQL query:

```
SELECT DISTINCT ?movie ?label ?year WHERE {
  ?movie rdf:type dbpedia-owl:Film.
  ?movie rdfs:label ?label.
  ?movie dcterms:subject ?cat .
  ?cat rdfs:label ?year .
  FILTER langMatches(lang(?label), "EN") .
  FILTER regex(?year, "^[0-9]{4} film", "i")
}
ORDER BY ?label
```

Then, we performed a one-to-one mapping with the movies in MovieLens by using the Levenshtein distance and checking the year of production. We found that 298 out of 3,952 (7.54%) movies in MovieLens have no correspondence DBpedia. After this automatic check we manually double-checked the results and we found that 69 out of 3,654 mappings (1.89%) were not correct and we manually fixed them. A dump of the mapping is available at the address: http://sisinflab.poliba.it/mapping-movielens-dbpedia-1M.zip. Before starting our evaluation, we had to align also the user profiles in MovieLens with the user profile in Equation (1). Indeed, in MovieLens user u expresses a rate on a movie m_j based on a five-valued scale: $r(u, m_j) \in [1, \ldots, 5]$, where a vote of 1 indicates an *Awful* movie, while a vote of 5 classifies a movie as *Must See*. In order to map the five stars rating to a binary rating, we computed for each user u in the dataset their average rating $\hat{r}(u)$ and we built user profile as

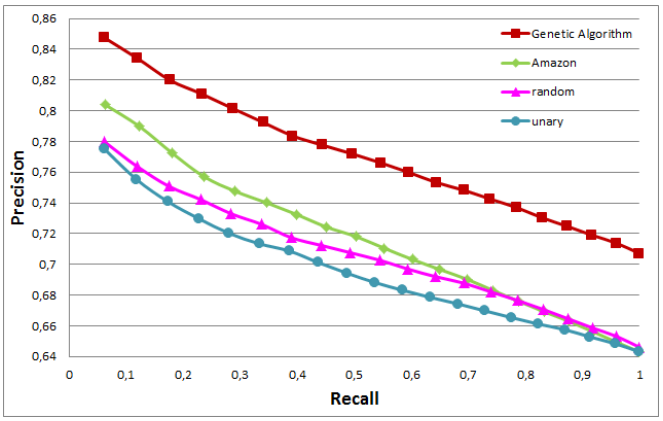
$$profile(u) = \{\langle m_j, v_j \rangle \mid v_j = 1 \text{ if } r(u, m_j) \geq \hat{r}(u), \; v_j = -1 \text{ otherwise}\}$$

The same consideration holds when we evaluate the recommendation algorithm in terms of precision and recall. In recommender systems, precision and recall are defined respectively as: *precision*: fraction of the top-N recommended items that are relevant to u; *recall*: fraction of the relevant items that are recommended to u. In our experiments, since we focus on the test set to find the actual relevant items of the target user, the top-N list we compute contains the items that are in the target user's test set only. The test set we extracted from the 1M MovieLens dataset contains 20 rates per user. For this reason, we computed the Precision@N and Recall@N, varying N in the interval $[1, \ldots, 20]$. More precisely, our experiments of Precision and Recall aim to evaluate the following different aspects: (i) comparing Equation 2 and Equation 3 to select the best performing one (Table 1); (ii) comparing different algorithms used for the selection of the α_p coefficients (Figure 3(a)); (iii) analyzing how choosing specific sets of properties and data affects the quality of results (Figure 3(b)); (iv) validating our approach by comparison with keyword-based and collaborative recommender systems (Figure 3(c)). To ensure the results are not biased by some user profiles, we executed a 5-fold cross-validation [20].

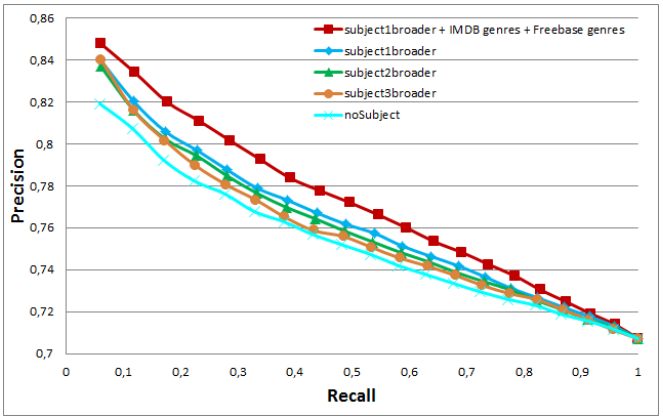
From Table 1 we see that Equation 2 and Equation 3 have comparable performance and Equation 2 performs slightly better than the other one. Hence, we continued our evaluation by considering only this one.

[2] http://wiki.dbpedia.org/Downloads37#ontologyinfoboxproperties

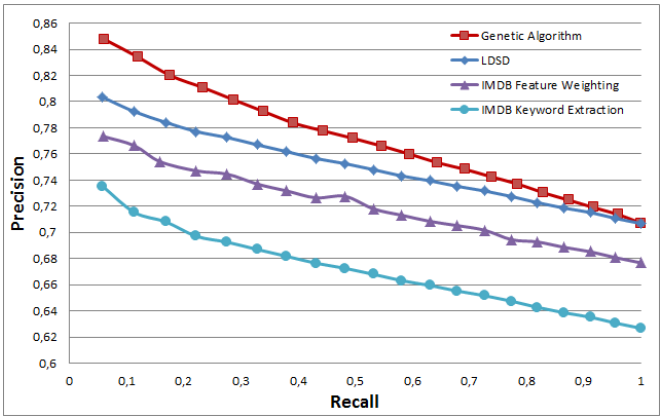
[3] It was not necessary to map also movies in LinkedMDB and Freebase since they are linked to DBpedia via owl:sameAs statements.



(a)



(b)



(c)

Figure 3: Precision and Recall curves obtained: (a) with different algorithms for computing the α_p coefficients; (b) considering different sets of LOD properties; (c) comparing our system with other content-based RSs.

	Equation 2		Equation 3	
N	Precision	Recall	Precision	Recall
1	0,834	0,060	0,831	0,060
3	0,806	0,174	0,802	0,173
5	0,788	0,282	0,786	0,282
10	0,757	0,540	0,751	0,537
15	0,731	0,778	0,727	0,777
20	0,707	1,000	0,704	1,000

Table 1: Precision and Recall evaluation for Equation 2 and Equation 3.

Figure 3(a) shows the Precision and Recall curves obtained considering different approaches for the computation of the α_p coefficients. In particular, for the curve identified by the square markers, the weights α_p computed via the genetic algorithm are used, as detailed in Section 3.1.1. The green curve with diamond markers refers to the coefficients calculated via statistical analysis on `Amazon` recommender system, as described in Section 3.1.2. Then, the last two curves show the results of Precision and Recall when the coefficients are randomly selected and when they are all set equal to 1, respectively. For the violet curve with triangle markers, we ran the random selection 60,000 times (about 10 times the number of users in the `MovieLens` dataset) and then we computed the mean values. For an infinite number of trials, the curve obtained with random selection should be identical to the curve where the coefficients are all equals. This is the reason why the two curves in the figure are very similar. It is evident from Figure 3(a) that the genetic algorithm allows the system to achieve the best results in terms of Precision. For this reason, in the following evaluation we rely on the α_p coefficients computed by this algorithm.

The following part of our evaluation aims to highlight the importance of the ontological information contained within `Linked Open Data` and in particular the categories available in `DBpedia`. To this purpose, we computed the recommendation lists for users in the test set (via Equation 2), considering specific groups of properties. The goal is to understand how these properties affect the quality of the recommendation. The results of Precision and Recall are shown in Figure 3(b). The red curve with square markers considers all the properties we extracted from the `DBpedia` ontology and the genres extracted from `LinkedMDB` and `Freebase`. In this curve, also the categories not directly related to a movie are considered – i.e., considering `skos:broader` as *one-step transitive* (see the highlighted path in Figure 1). The best results are achieved in this case. The curve *subject1broader* does not consider information coming from `Freebase` and `LinkedMDB`. We may see how the results get worse thus showing that information coming from different datasets is meaningful and does not add noise to data. We also evaluated the system by considering different "*degree of transitivity*" for `skos:broader` (see the curves *subject2broader* and *subject3broader*). From Figure 3(b) it is possible to observe that in these cases the results get worse and worse. This means that more generic categories are not so meaningful and add noise to the information. It is interesting to note that the IDF does not mitigate the importance of these categories within the data. We expected that the more generic a category is, the more frequently it appears. Finally, the cyan curve with cross markers does not consider any `DBpedia` category nor `IMDB`/`Freebase` genre, but only the properties in

the DBpedia ontology. We may say that the information contained in the categories one step far from a movie is very important for the quality of the recommendation. We stress that such type of information can be found only in ontological datasets as the ones belonging to Linked Open Data cloud.

In the last part of the evaluation of our system, we focus on the comparison with a related approach based on DBpedia and with some keyword-based approaches (see Figure 3(c)). The curve with square markers refers to the algorithm used in our system, where the α_p coefficients are obtained via a genetic algorithm as detailed in 3.1.1 and all the properties within the semantic graph are considered. In particular, skos:broader is considered as *one-step* transitive. All the other curves refer to related work about some content-based recommender systems. The blue curve with diamond markers refers to LDSD (Linked Data Semantic Distance) [18], a measure applied to build a music RS. We re-implemented the algorithms proposed by the authors and made a comparison with our approach. The graph shows that our recommender system performs better in the movie domain. The curve represented by triangle markers refers to the work by Debnath et al. [7]. Here, the authors leverage structured data coming from IMDB to produce a content-based recommendation. They estimate the values of the weights for the properties involved in the recommendation by a set of linear regression equations obtained from a social network graph. Lastly, the curve indicated with circle markers refers to a work by Wartena et al. [26], and it can be seen as a baseline approach to CB recommendation. The authors augment the MovieLens dataset with IMDB textual description about movies. Then, keywords are extracted from text and movies are described by bags of words. The main drawback of this approach is that the informative sources considered are not structured. This is the reason why all the other approaches we compare, being based on structured data, obtain better accuracy results measured by Precision/Recall. Thanks to the ontological information within the Linked Open Data datasets we exploit, we are able to improve the quality of a standard content-based system based on structured data.

5. RELATED WORK

At the best of our knowledge, our system is one of the very first initiatives that leverage Linked Open Data to build recommender systems. A lot of systems have been proposed in literature that address the problem of recommendations, but there are very few approaches that exploit the data within LOD to provide recommendations. In the following we give a brief overview of semantic-based approaches to recommendation. Similarly to our approach, in [28] the authors rely on structured features to show their superiority with respect to a pure term-based similarity computation. However, at the time of that work, LOD was not available yet. *dbrec* [17] is a music content-based recommender system leveraging the DBpedia dataset. The approach is link-based, i.e. the "semantics" of relations is not exploited since each relation has the same importance, and it does not take into account the links hierarchy, expressed in DBpedia through the DCTERMS and SKOS vocabulary. However, the properties within these vocabularies have proven to be fundamental in our evaluation (cf. Figure 3(b)). Szomszor et al. [24] investigate the use of folksonomies to generate tag-clouds that can be used to build better user profiles to enhance the movie recommendation. They use an ontology to integrate both IMDB and Netflix data. However, they compute similarities among movies taking into account just similarities between movie-tags and keywords in the tag-cloud, without considering other information like actors, directors, writers as we do here. *Filmtrust* [11] integrates Semantic Web-based social networking into a movie recommender system. Trust has been encoded using the FOAF Trust Module and is exploited to provide predictive movie recommendation. It uses a collaborative filtering approach as many other recommender systems, as *MovieLens* [14], *Recommendz* [10] and *Film-Consei* [19]. Our RDF graph representation as a three-dimensional tensor has been inspired by [9]. Here, the authors extend the paradigm of two-dimensional graph representation to obtain information on resources and predicates of the analyzed graph. In the pre-processing phase they just prune dominant predicates, while we automatically extract, through SPARQL queries, relevant ones according to the chosen domain. Moreover, in [9] the authors consider only the objects of the RDF triples, while we look also at subject of the statements to compute similarities. Tous and Delgado [25] use the vector space model to compute similarities between entities for ontology alignment, however with their approach it is possible to handle only a subset of the cases we consider, specifically only the case where resources are directly linked. Eidon et al. [8] represent each concept in an RDF graph as a vector containing non-zero weights, but they take into account only the distance from concepts and the sub-class relation to compute such weights. Heitmann and Hayes [13] propose to leverage Linked Open Data to build open recommender systems. The purpose is to mitigate the new-user, new-item and sparsity problems of collaborative recommender systems. As for [17], the domain is the music. Differently from our approach, they do not exploit any ontological information of DBpedia. Moreover, being based on collaborative-filtering, their aim is to fill a user-item matrix with binary values, while we define P vector spaces – P is the number of the properties considered in the recommendation (cf. Section 2.1) – to compute the similarity between resources in order to build a content-based recommender. In [22] the authors propose to use Linked Open Data for educational purposes. The presented system is a *Resource List Management System* built on LOD principles. However, the recommender part is just a vision.

6. CONCLUSION AND FUTURE WORK

The usage of Linked Open Data datasets poses new challenges and issues in the development of next generation recommender system and, more generally, complex Web applications. In this paper we have presented a content-based recommender system that leverages the knowledge encoded in semantic datasets of the Linked Open Data project. In particular, since we focused on the movie domain, we exploited the data within DBpedia, Freebase and LinkedMDB to collect information about movies, actors, directors, genres, categories, etc.. The innovative aspect of the research is the usage of Linked Open Data as the only background knowledge of a movie recommender system, and the adaptation of a popular ranking measure in Information Retrieval like the Vector Space Model to semantic networks. Currently, we are working on testing/integrating other approaches for the recommendation (e.g., *Support Vector Machine*) applied to Linked Open Data. We will further expand the collec-

tion of LOD datasets used in the recommendation, and we will extend the whole approach to other domains, to eventually propose a cross-domain recommendation leveraging LOD datasets. Moreover we want to combine a LOD-based recommendation with a collaborative-filtering approach in order to improve the performances of our system. Finally, we are evaluating how LOD-based recommender systems may mitigate common issues related to *Limited Content Analysis*, cold-start and sparsity of data.

Acknowledgments

The authors wish to thank Marco de Gemmis for fruitful discussions and suggestions. This research is partially sponsored by HP IRP 2011. Grant CW267313.

7. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge Data Engineering*, 17(6):734–749, 2005.

[2] R. A. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval: The Concepts and Technology behind Search*. Addison-Wesley Professional, 2011.

[3] T. Berners-Lee, Y. Chen, L. Chilton, D. Connolly, R. Dhanaraj, J. Hollenbach, A. Lerer, and D. Sheets. Tabulator: Exploring and analyzing linked data on the semantic web. In *In Procedings of the 3rd International Semantic Web User Interaction Workshop (SWUI06)*, 2006.

[4] C. Bizer, T. Heath, and T. Berners-Lee. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst*, 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. *Web Semant.*, 7:154–165, September 2009.

[6] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD '08*, pages 1247–1250. ACM, 2008.

[7] S. Debnath, N. Ganguly, and P. Mitra. Feature weighting in content based recommendation system using social network analysis. In *Proceedings of the 17th international conference on World Wide Web*, WWW '08, pages 1041–1042, New York, NY, USA, 2008. ACM.

[8] Z. Eidoon, N. Yazdani, and F. Oroumchian. A vector based method of ontology matching. In *Proc. of 3rd Int. Conf. on Semantics, Knowledge and Grid*, pages 378–381, 2007.

[9] T. Franz, A. Schultz, S. Sizov, and S. Staab. Triplerank: Ranking semantic web data by tensor decomposition. In *Proc. of the 8th ISWC*, ISWC '09, pages 213–228, 2009.

[10] M. Garden and G. Dudek. Semantic feedback for hybrid recommendations in recommendz. In *IEEE Int. Conf. EEE'05*, pages 754–759, 2005.

[11] J. Golbeck and J. Hendler. Filmtrust: Movie recommendations using trust in web-based social networks. In *Proceedings of the IEEE CCNC*, 2006.

[12] O. Hassanzadeh and M. P. Consens. Linked Movie Data Base. In *Proceedings of the WWW2009 Workshop on Linked Data on the Web*, April 2009.

[13] B. Heitmann and C. Hayes. Using linked data to build open, collaborative recommender systems. In *AAAI Spring Symposium: Linked Data Meets Artificial Intelligence*, 2010.

[14] J. Herlocker, J. A. Konstan, and J. Riedl. Explaining collaborative filtering recommendations. In *Proceeding on the ACM 2000 Conference on Computer Supported Cooperative Work*, pages 241–250, 2000.

[15] P. Lops, M. de Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender Systems Handbook*, pages 73–105. 2011.

[16] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[17] A. Passant. dbrec: music recommendations using dbpedia. In *Proc. of 9th Int. Sem. Web Conf.*, ISWC'10, pages 209–224, 2010.

[18] A. Passant and S. Decker. Hey! ho! let's go! explanatory music recommendations with dbrec. In *ESWC (2)*, pages 411–415, 2010.

[19] P. Perny and J. Zucker. Preference-based search and machine learning for collaborative filtering: the film-consei recommender system. *Information, Interaction, Intelligence*, 1:9–48, 2001.

[20] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors. *Recommender Systems Handbook*. Springer, 2011.

[21] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18:613–620, November 1975.

[22] N. Shabir and C. Clarke. Using linked data as a basis for a learning resource recommendation system. In *Learning in the Synergy of Multiple Disciplines, Proceedings of EC-TEL'09*, 2009.

[23] M. Srinivas and L. Patnaik. Genetic algorithms: a survey. *Computer*, 27(6):17 –26, jun 1994.

[24] 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 *4th ESWC*, 2007.

[25] R. Tous and J. Delgado. A vector space model for semantic similarity calculation and owl ontology alignment. In *DEXA*, pages 307–316, 2006.

[26] C. Wartena, W. Slakhorst, and M. Wibbels. Selecting keywords for content based recommendation. In *CIKM*, pages 1533–1536, 2010.

[27] R. W. White and R. A. Roth. Exploratory Search: Beyond the Query-Response Paradigm. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 1(1):1–98, Jan. 2009.

[28] M. Zanker, S. Gordea, M. Jessenitschnig, and M. Schnabl. A hybrid similarity concept for browsing semi-structured product items. In *EC-Web*, volume 4082 of *Lecture Notes in Computer Science*, pages 21–30. Springer, 2006.