Expert Systems with Applications 39 (2012) 10990–11000

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Social knowledge-based recommender system. Application to the movies domain

Walter Carrer-Neto, María Luisa Hernández-Alcaraz, Rafael Valencia-García *, Francisco García-Sánchez

Departamento de Informática y Sistemas, Universidad de Murcia, 30100 Murcia, Spain

ARTICLE INFO

Keywords:
Recommender systems
Ontologies
Semantic Web
Knowledge-based systems

ABSTRACT

With the advent of the Social Web and the growing popularity of Web 2.0 applications, recommender systems are gaining momentum. The recommendations generated by these systems aim to provide end users with suggestions about information items, social elements, products or services that are likely to be of their interest. The traditional syntactic-based recommender systems suffer from a number of shortcomings that hamper their effectiveness. As semantic technologies mature, they provide a consistent and reliable basis for dealing with data at the knowledge level. Adding semantically empowered techniques to recommender systems can significantly improve the overall quality of recommendations. In this work, a hybrid recommender system based on knowledge and social networks is presented. Its evaluation in the cinematographic domain yields very promising results compared to state-of-the-art solutions.

1. Introduction

In today's society, recommendations are becoming increasingly important and users use them to identify products, services or contents they might like. That is why, in recent years, many companies and Web sites have implemented systems to study the preferences of their users in order to direct products/information/services to them in a more accurate manner, therefore improving the company's profits. Recommender systems have been applied to several different domains such as tourism/leisure (hotels, restaurants, parks, beaches, etc.) (Stanley, Lorenzi, Saldaña, & Licthnow, 2003), advertising (Cheung, Kwok, Law, & Tsui, 2003), ecommerce (Ghani & Fano, 2002), film/TV/music, among others. However, due to the incredible growth of the Web in the recent years, searching and finding products, services or content that may be of interest for users has become a very tough task. This happens because the information available on the internet has been designed to be readable only by humans, and so computer systems can not process nor interpret the data present in it.

The Semantic Web technologies and concepts have emerged to overcome these deficiencies. The Semantic Web has revolutionized the way that systems integrate and share data, enabling computational agents to reason about information and infer new knowledge (Berners-Lee, Hendler, & Lassila, 2001) becoming a very promising platform for developing knowledge management systems. Ontologies (Studer, Benjamins, & Fensel, 1998) constitute the standard knowledge representation mechanism for the Semantic Web. The formal semantics underlying ontology languages enables the automatic processing of the information in ontologies and allows the use of semantic reasoners to infer new knowledge. In this work, an ontology is seen as "a formal and explicit specification of a shared conceptualization" (Studer et al., 1998). Knowledge in ontologies is mainly formalized using five kinds of components: classes, relations, functions, axioms and instances (Gruber, 1993).

The Ontology Web Language (OWL) is the W3C standard for representing ontologies in the Semantic Web and, in this work, it has been used to represent the knowledge that is handled by the recommender system proposed here.

Ontologies are thus the key for the success of the Semantic Web vision. The use of ontologies can overcome the limitations of traditional natural language processing methods and they are also relevant in the scope of the mechanisms related, for instance, with Information Retrieval (Valencia-García, Fernández-Breis, Ruiz-Martínez, García-Sánchez, & Martínez-Béjar, 2008), Semantic Search (Lupiani-Ruiz et al., 2011), Service Discovery (García-Sánchez, Valencia-García, Martínez-Béjar, & Fernández-Breis, 2009), Question Answering (Valencia-García, García-Sánchez, Castellanos-Nieves, & Fernández-Breis, 2011), searching for contents and information (Jiang & Tan, 2009), as well as crawling (Yang, 2009). On the other hand, ontologies have been applied to numerous domains such as tourism (Jung, 2011), biomedicine (Arsene, Dumitrache & Mihu, 2011; Ruiz-Martínez, Valencia-García, Fernández-Breis, García-Sánchez & Martínez-Béjar, 2011) or ubiquitous computing (Yang, 2011). In fact, recommender systems have already begun to

* Corresponding author. Fax: +34 868884151.
E-mail addresses: carrer@gmail.com (W. Carrer-Neto), mlhernandez@um.es (M.L. Hernández-Alcaraz), valencia@um.es (R. Valencia-García), frgarcia@um.es (F. García-Sánchez).

0957-4174/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2012.03.025

employ ontologies in order to improve theirs mechanisms of recommendation (Blanco-Fernández et al., 2008a; Yang, 2010).

In this paper, we present a social-based contents recommender system that makes use of the Semantic Web principles to build a mechanism to help users find content that is relevant in accordance with their preferences. Moreover, we propose an alternative way of using a contemporary resource, namely social networks, to build a novel way of collaborative filtering.

The remainder of this paper is structured as follows. Section 2 shows an overview of the different approaches for designing recommender systems. In Section 3, the system developed in this work is described. Section 4 explains the experiments conducted to evaluate the proposed framework. Finally, some conclusions and future work are put forward in Section 5.

2. Related work. Recommender systems

Recommender systems began to appear in the market in 1996 (Manber, Patel, & Robison, 2000). In the context of recommender systems design and implementation, four different approaches can be distinguished (Adomavicius & Tuzhilin, 2005; Burke, 2000):

- Content-based recommender systems: they try to find products, services or contents that are similar to those already evaluated by the user. In this kind of systems, user's feedback (that can be collected in many ways) are essential to support and accomplish recommendations.
- Knowledge-based recommender systems: they model the user profile in order to, through inference algorithms, identify the correlation between their preferences and existing products, services or content.
- Collaborative filtering recommender systems: they create/classify groups of users that share similar profiles/behaviors in order to recommend products, services or content that has been well evaluated by the group to which a user belongs.
- Hybrid recommender systems: they combine two or more techniques to improve the "quality" of recommendations.

Pure content-based systems suffer from what it is known as overspecialization, that is, to specialize to an extreme degree. By trying to find contents similar to the ones users have already seen, the system tends to end up recommending the same group of contents (Adams, Bennett, & Tomasic, 2007). Since each technique has its own pros and cons, most solutions combine different techniques so facilitating the improvement of their overall performance.

On the other hand, knowledge-based systems are becoming very important. For example, AVATAR (Blanco-Fernández et al., 2008a) is a recommender system for TV content that uses Semantic Web technologies to produce a knowledge-based system that allows the system to infer and reason about ontologies, a well known and powerful tool for intelligent inference.

Systems that make use of collaborative filtering have existed for a long time, such as the well-known Tapestry (Goldberg, Nichols, Oki, & Terry, 1992), a system developed to help users to "filter" electronic documents received by e-mail; Ringo (Shardanand & Maes, 1995), a recommender system for music; GroupLens (Resnick, Iacovou, Suchak, Bergstrom, & Riedl, 1994), a recommender system for Usenet articles/messages, MovieLens (Dahlen et al., 1998); or PHOAKS (Terveen, Hill, Amento, McDonald, & Creter, 1997), a recommender system for links/contents, among others.

Collaborative filtering-based systems propelled commercial services such as Amazon,[1] CDNow[2] and Netflix.[3] In fact, Netflix has invested so much in recommender systems that in 2006, launched a competition of US$ 1,000,000 to the first person who managed to improve their recommender system by a 10%.[4]

Nowadays hybrid systems are gaining momentum. For instance, Recommendz (Garden & Dudek, 2005), a movie recommender system, makes use of semantics in its content-based module together with a collaborative filtering technique. This system makes fairly interesting use of "semantic feedback" from users inside the recommender engine. FOAFing-the-music (Celma, 2006), a hybrid system that is entirely based on trust in social networks, also uses semantic analysis of contents.

Trust is a fundamental requirement for any recommender system. For example, the mechanism used in FilmTrust (Golbeck & Hendler, 2006) is based entirely on trust in social networks. Other works such as Sinha and Swearingen (2001), Herlocker, Konstan, and Riedl (2000) and Swearingen and Sinha (2001) show the importance of trust on the recommendations and present guidelines that recommender systems should follow to accomplish this requirement. Finally, Massa and Avesani (2004) shows how to use the explicit statements of trust between users on a network within a collaborative filtering recommender system.

There are a number of issues that hamper the functioning of recommender systems. For example, the cold start problem (Lam, Vu, Le, & Duong, 2008) represents a difficulty that these systems face when trying to provide recommendations to new users and about new contents. This problem is mainly due to the lack of information related to those users/contents. Until users add up a significant amount of information about theirs preferences, pure content-based systems fail to effectively provide recommendations to them. On knowledge-based systems, even though they take into account the main characteristics of contents, this kind of problem might still occur if the users do not enter a good description of their preferences. On the other hand, collaborative filtering systems help to alleviate the lack of information by inheriting recommendations from other users. The scarcity of data also affects collaborative filtering systems in such a way that if a content is not evaluated by a representative number of users from the network, it will most likely not be frequently recommended. In Yildirim and Krishnamoorthy (2008) and Bogers (2010), the authors propose a way to smooth down this problem by passively getting feedback from the way users navigate through the contents (by checking the web-pages they have visited): On the other hand, in Amatriain, Lathia, Pujol, Kwak, and Oliver (2009), the authors present a way to prioritize the "voice" of a minority of users, who represent a group of experts in a specific field, making their classification about contents more significant to the system.

Fake profiles can constitute a further problem to social networks-based collaborative filtering systems. These profiles can affect some contents' rating in order to promote their popularity, thus "forcing" the system to suggest a especial set of items. In Williams, Bhaumik, Sandvig, Mobasher, and Burke (2008), the authors propose a method for detecting such fictitious users in a way that their ratings are not taken into account.

Recommender systems that use collaborative filtering aim at grouping together users in according to their preferences in order to share recommendations among them. Semantic knowledge-based systems can reinforce the recommendations since they do not base their suggestions on this filter, but on the semantic similarity between the contents and the users' preferences.

In this work, we present a hybrid recommender system based on knowledge and social networks. In the proposed system, users are free to choose the factor by which their social network will

[1] http://www.amazon.com/.
[2] http://www.cdnow.com/.
[3] http://www.netflix.com/.
[4] http://www.digitaltrends.com/entertainment/netflix-gives-1-million-prize-to-bellkors-pragmatic-chaos/.

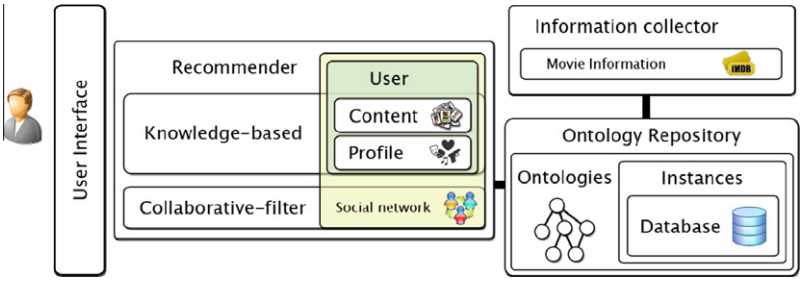

Fig. 1. Architecture of the system.

influence on the recommendations directed to them. Thus, some users may adopt a conservative posture in which the system does not get any recommendation of their network, while other users with a more open-minded profile may be willing to inherit recommendations based on the preferences of their friends. The work presented in this paper has been developed in the cinematic context for it be a well-known domain and because of the wide number of elements and relationships that exist between the concepts involved in the system (such as movies, actors, directors, genres, users, etc.). Another reason for validating this tool in the movie domain is the existence of ontologies (e.g. Movie Ontology) and sources of information (e.g. IMDB) that can be used to populate those ontologies.

3. Social knowledge-based recommender system

In this section, the technologies that have been combined to build the semantic recommender system are presented. Fig. 1 illustrates the architecture of the proposed system. In a nutshell the systems works as follows. Users manage their profile preferences (1) by adding and classifying elements (contents or characteristics) according to their personal preferences, and (2) by creating friendship links with other users. Then, the system, through the information collector module, gathers data on demand from the IMDB website and instantiates the ontology.

Finally, the recommender module is in charge of analyzing the likings, preferences and social network of users, and suggesting matching movies that they might like.

As illustrated in Fig. 1, our framework is composed of three modules, namely, ontology repository, data collector and the recommender system. The following sections describe each of these modules in detail.

3.1. Ontology repository

Finding an efficient and effective way to store and manage the ontologies and their elements is something crucial to the proposed framework given the large amount of information that must be processed. The features that should be considered when choosing a persistent storage system over others range from the existence of APIs, supported query languages, performance (speed of access, reading/writing, search, scalability), supported inference mechanisms to, of course, costs (freely available or commercial solutions).

Most of the current ontology repositories work on the top of relational databases such as MySQL and PostgreSQL to provide data persistence. Despite the fact that there are several tools available (e.g., Sesame, OWLIM, OWLDB, Mulgara, Redland, KAON, etc.), in our previous experiments we witnessed some sort of inefficiency in the open source ontology repository systems, as well as a lack of scalability to deal with millions of data items. For this reason, we decided to develop a simple and optimized storage system based on relational databases and OWL files.

Relational databases are more efficient (in terms of writing/reading/searching data) than most ontology repositories and so, in addition to storing user data and social network related information, the framework makes also use of a relational database to store the ontology and its instances. More specifically, the system database stores the description of classes, properties/relationships and existing individuals/resources.

3.1.1. The Movie Ontology

The Movie Ontology[5] aims to provide a controlled vocabulary to describe the semantic concepts related to the movies domain such as Gender, Director, Actor and its individuals such, for example, "Western", "Comedy", "Inglourious Basterds", "Steven Spielberg" or "Angelina Jolie". It has been developed according to the OWL standard by the University of Zurich.

Through this ontology it is possible to link, hierarchically and semantically, elements belonging to the movies domain. The main class of the ontology is the class Movie. All the films are instances/individuals of this class. The main datatype and object properties that are used in the system presented here are the following:

- belongsToGenre: an object property that represents the genres a movie is associated with. The class Genre has several hierarchical levels or, in other words, subclasses such as "Comedy", "Drama" and "Western". This relationship is inverse to isGenreOf relationship.
- hasActor, hasDirector, hasProducer: these relationships are linked to the class Person. The hierarchical level of Person's subclasses specifies the role that a person plays (i.e. Actor, Director, Producer, etc.). These isDirectorOf and isActorInrelationships are the inverse of the hasDirector and hasActor relationships respectively, while hasProducer has no inverse relationship.
- isAwardedWith, nominatedFor: object properties that list the awards a movie has won and the ones for which it has been nominated, respectively. isAwardOf is the inverse of isAwardedWith.
- hasFilmLocation: this object property defines the locations related to the filming. Its inverse is isFilmLocationOf.
- isProducedBy: object property that relates a film with its production company (e.g. "Walt Disney Pictures", "DreamWorks", "Paramount Pictures", etc.). Its inverse is produced.
- isFromDecade: this object property classifies the film by its age, representing the decade in which it was launched. released is the inverse of this relationship.
- isTranslatedTo: describes the languages to which a film was translated. Its inverse is hasTranslatedMovie.

For the purposes of our work, the Movie Ontology has been slightly modified. The main changes are (1) the inclusion of the attribute imdbcode to the Movie class, which obviously represents the film's code on the IMDB database, and (2) the creation of the

[5] http://www.movieontology.org/.

class Age, which represents the decade a film belongs to by means of the aforementioned released relationship.

3.2. Information collector

In order to recommend movies it is necessary to extract data bout such movies from a real source and to update the domain ontology with this information. This process is called ontology population and is responsible for extracting and classifying the instances of the concepts and relations that have been defined in the ontology (Ruiz-Martínez, Miñarro-Giménez, Castellanos-Nieves, García-Sánchez, & Valencia-Garcia, 2011). Instantiating ontologies with new knowledge is a relevant step towards the provision of valuable ontology-based knowledge services.

We can distinguish two types of ontology population: (i) ontology population from free text (see for example Ruiz-Martínez et al. (2011)), and (ii) ontology population from semi-structured documents (e.g., XML, HTML, etc.). In this work, a semiautomatic method for ontology population from structured data based on previous work (García-Manotas, Lupiani, Garcıa-Sánchez, & Valencia-García, 2010) has been developed.

The Internet Movie Database (IMDB[6]) represents one of the largest internet sources for movies/actors data. For the purposes of our use case evaluation, this portal has been used as the data source for populating the domain ontology. For this, the IMDBPHP[7] API has been used in order to serialize and package all the data related to a movie in an XML file.

3.3. Recommender system

The recommendation module is a hybrid tool consisting of a knowledge-based subsystem and a collaborative filtering service feeding from the users' relationships through their social network. In the following sections, the way these modules operate is described in detail.

3.3.1. Knowledge-based recommender

The knowledge-based recommender system is the main part of the system and tries to recommend similar instances from the ones that match the users' preferences of the "reference class" of the domain ontology. In our use case scenario, the domain ontology is related to the movies domain and the "reference class" is the class "Movie". Therefore the system tries to recommend movies based on the user preferences and likings.

To achieve its aims, the recommender system maintains an updated matrix of order n. This matrix contains the similarity rates between all the instances (n instances) that belong to the "reference class", which represents the "target" of the recommender engine (in this case, the Movie class). This matrix is called Similarity Reference Matrix (see Fig. 2) and each element *Xij* is a float number that represents the similarity between the individuals *i* and *j*. From now on, this matrix will be referred to as RefSim.

Each element of the RefSim matrix is calculated according to the Formula (1), which is based on the formula SemSim presented in Blanco-Fernández et al. (2008b). For the purposes of our use case, the hierarchical semantic similarity is not considered to calculate similarities between individuals, since there is one single hierarchical level (class Movies).

$$S_{ab} = \text{Similarity}(a,b) = \sum_{i=1}^{\#P}\left(\frac{\text{common}(a,b,P[i])}{\max(\deg(a,P[i]),\deg(b,P[i]))}\right) * \text{Weight}(P[i]) \quad (1)$$

where

$$RefSim = \begin{bmatrix} 1 & S_{12} & S_{13} & \dots & S_{1n} \\ S_{21} & 1 & S_{23} & \dots & S_{2n} \\ S_{31} & S_{32} & 1 & \dots & S_{3n} \\ \dots & \dots & \dots & 1 & \dots \\ S_{n1} & S_{n2} & S_{n3} & \dots & 1 \end{bmatrix} \text{ where,}$$

- $S_{ab} = S_{ba} =$ similarity between A and B; $S_{cc} = 1$;

Fig. 2. Similarity reference matrix (RefSim).

- P (i.e., Properties) is a vector that contains a set of attributes (datatype properties) and relationships (object properties) of the reference class that the formula should take into account when calculating similarities. For example:
  - P = [hasColor, belongsToGenre, isFromDecade, hasFilmLocation], #P = 4.
  - P = [hasActor, imdbrating], #P = 2.
- deg(a,p) represents the number of instances associated to the individual a through the relation p. For example:
  - A.belongsToGenre = [Drama, Horror], degree (A,belongsToGenre) = 2.
- common(a, b, p) represents the number of common instances associated to a and b through the property or relation p. For example:
  - A.belongsToGenre = [Drama, Horror].
  - B.belongsToGenre = [Romance, Drama, Sci-fi], common(A, B, belongsT oGenre) = 1.
- Weight(p) expresses the importance (by means of "weight") of the property or relation p in the formula. For example:
  - Weight(belongsToGenre) = 0.2.

One can observe that it is possible to weigh each of the class' properties to have more importance/influence in the calculation of the similarity between individuals. For example, the Genre of a movie is more important than its filming locations. These weights can and should be set according to the domain in which the framework is being applied.

The formula (1) can be implemented according to the pseudocode presented in Algorithm 1 (see Fig. 3).

Once implemented the similarity formula, it is possible to generate the RefSim matrix through Algorithm 2 (see Fig. 4).

It is very likely that many of the elements of the RefSim matrix would get a very low rate indicating that these elements do not have much in common. Therefore, the system assumes that the similarity between two elements is meaningful only if it exceeds a certain threshold.

In order to calculate this threshold it is possible to employ statisticals methods, such as the use of the concept of percentiles (deciles, quartiles, etc.) on a sample of the generated rates. The percentile is the value of a variable (in this case, the similarity rates) below which a certain percent of observations fall. Using this concept, the system can estimate the threshold value as the rate that separates the 90% of the levels of similarity, represented by P90. That is, the value that cuts apart 90% of the samples, letting the other 10% representing meaningful similar items. Percentiles can be calculated through the Formula (2) presented in Mendenhall and Sincich (2006).

$$k = \frac{p * n}{100} \quad (2)$$

where

1. k is the position corresponding to the value that separates the sample (where the values are sorted in ascending order) in p percentile;
2. p represents the percentile we're looking for;
3. n represents the sample size;

[6] http://www.imdb.com/.
[7] http://projects.izzysoft.de/trac/imdbphp.

Algorithm 1: *Similarity* function

Input: A and B represent sets of instances where $\{\{E_1, E_2, E_3, ..., E_n\}|E_x$ is a set of elements of type $x\}$; Set Weight $=\{(r_1, k_1), (r_2, k_2), (r_3, k_3), ..., (r_m, k_m)|r_x$ is a property/relationship and k_x its weight in the equation$\}$.
Output: Similarity rate between A and B normalized in [0,1]
subTotal $\leftarrow 0$
totalWeight $\leftarrow 0$
foreach $(r_i, k_i) \in$ Weight **do** // for each element of Weight
  // r_i is an property/relationship, i.e.: 'hasFilmLocation'
  // k_i is r_i's weight, i.e.: 0.3
  elementsA $\leftarrow E_{r_i} \in A$ // elements related to A through r_i
  elementsB $\leftarrow E_{r_i} \in B$ // elements related to B through r_i
  totalWeight $\leftarrow$ totalWeight $+ k_i$
  common $\leftarrow$ elementsA $\cap$ elementsB // elements (r_i) in common between A and B
  denominator $= max(\#$elementsA, $\#$elementsB$)$
  if denominator $\neq 0$ **then**
    subTotal $\leftarrow$ subTotal $+ \frac{\text{common} * k_i}{\text{denominator}}$
  end
end
return (subTotal/ totalWeight) // similarity index normalized in [0,1]

Fig. 3. Similarity function algorithm.

Algorithm 2: Algorithm that generates Similarity Reference Matrix *RefSim*

Input: Set of instances P $= (P_1, P_2, P_3, ..., P_n)$; Set Weight containing all properties/relationships and their weights that configures the equation
Output: Similarity Reference Matrix
for $i \leftarrow 1$ **to** $\#$P **do**
  for $j \leftarrow 1$ **to** $\#$P **do**
    if $i < j$ **then**
      RefSim$[i][j] \leftarrow$ Similarity$(P_i, P_j,$ Weight$)$
    end
  end
end
return RefSim

Fig. 4. RefSim matrix generator algorithm.

The system catalogs the recommendations that are to be made to a user through the Recon vector (see formula (3)). The indexes in the Recon vector represent movies and its values reflect the inferred degree of interest that a user might have about such movies. Therefore, the higher the value is, the greater the likelihood between the content and the user's interests.

$$recon_u = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ \dots \\ P_n \end{bmatrix} \quad (3)$$

where Pa = accumulated degree of interest inferred by comparing the profile of user u against the element (movie) a. The higher the value, the greater the relationship between the content and the user's preferences.

Users can classify any type of elements, which means that they can evaluate not only movies but also their characteristics (such as Gender, Actors, Awards and so on) rating individuals as [Horrible (−1), Very bad (−0.6), Bad (−0.3), Regular (0), Good (0.3), Very Good (0.6) and Great (1)].

The RefSim matrix might be used to adjust and calculate the vector Recon that, as already said, represents the matching degree between the contents (i.e., individuals from the reference class, Movie) and the user's preferences. The Recon vector contains probable recommendations that are to be made to a particular user, once the vector has been sorted in descending order.

In conclusion, each time users add information concerning their preferences (e.g., rate a movie or other property that they like or dislike), their corresponding Recon vector is adjusted as follows:

- If a user rates an element that belongs to the reference class, in other words, if a user rates a movie, the system, through the RefSim matrix, search for all other individuals that are similar to the referred item and adds or subtracts points from these movies in the Recon vector. The value added or subtracted would be directly related to the level of similarity between the contents.
- If a user rates elements from other classes (e.g., Actor, Genre, Director, etc.), or datatypes properties, the system searches for all the individuals belonging to the reference class (i.e., Movie) that are directly related to the rated item and adds or subtracts points from these movies in the Recon vector.

Fig. 5 illustrates how this process works and Algorithm (3) shows how the Recon vector is calculated and updated.

To help us illustrate how Algorithm 3 (see Fig. 6) works. Let us assume that a user A added three data items on his preferences information: (i) he thinks that the film Inglourious Basterds is "Very Good" (0.6), (ii) Quentin Tarantino, as director, is "Great" (1.0), but (iii) he does not like Uma Thurman (−1.0). When the system processes the first assertion (the one about Inglourious Basterds), given that it concerns a movie, the algorithm proceeds as follows:

- First, it searches for all individuals similar to this movie in the RefSim matrix. The first ten results would be (film [similarity index]): Kill Bill: Vol 2 [0.3421], Kill Bill: Vol 1 [0.3265] Pulp Fiction [0.3009], Master & Commander [0.2958], Crash [0.2934], Jack Brown [0.2876], Babel [0.2874], Requiem for a Dream [0.2847] Cast Away [0.2821] and Sin City [0.2819].
- Then, for each found movie, it increases its value in the Recon vector according to the classification assigned by the user, in

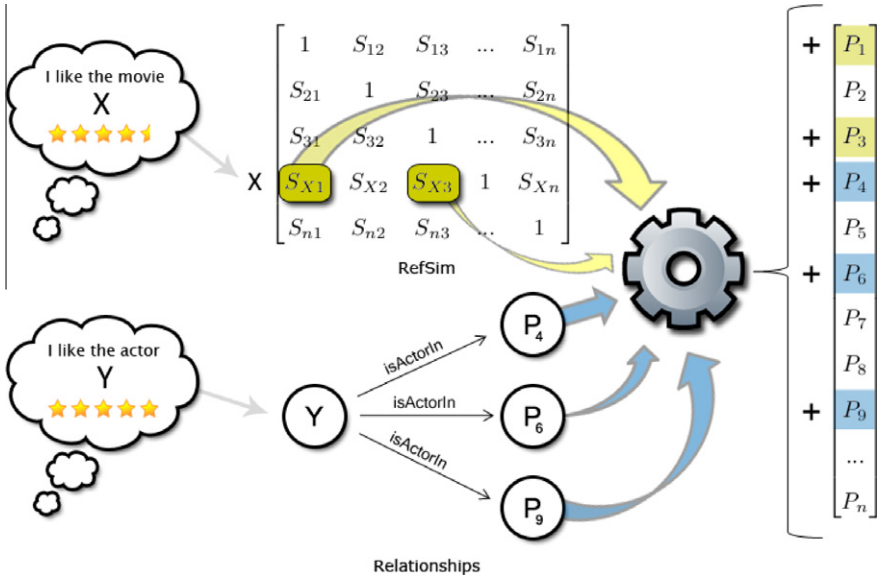

Fig. 5. Inferring user's preferences.

Algorithm 3: Inferring user's preferences (*Recon*)

Input: Set U containing all instances; Instance obj evaluated; Evaluation rate rate ([-1, -0.6, -0.3, 0, 0.3, 0.6, 1]); Similarity Reference Matrix RefSim; Set props that defines the relationships by which the system can associate instances in relationships, where $\text{relationships}_{\text{obj}}^{\text{props}}$ represents all elements belonging to the main class (*Movie*) that are associated to obj through these relationships; Vector recon;

Output: Vector recon updated

```
if belongsToMainClass (obj) then                    // it's a movie, belongs to the Movie class
    foreach x ∈ U|(obj, x) ∈ RefSim do                    // for each movie similar to obj
        recon[x] ← recon[x] + (rate * RefSim[obj][x])
    end
else                                                // it's not a movie, can be an actor, genre, etc
    foreach x ∈ U|x ∈ relationships_obj^props do          // for each movie related to obj
        recon[x] ← recon[x] + rate
    end
end
return recon
```

Fig. 6. Algorithm for inferring user's preferences.

proportion to their similarity value. Thus, the Recon vector would have the following values: $0.3421 * 0.6$, $0.3265 * 0.6$, $0.3009 * 0.6$, $0.2958 * 0.6$, $0.2934 * 0.6$, $0.2876 * 0.6$, $0.2874 * 0.6$, $0.2847 * 0.6$, $0.2821 * 0.6$, $0.2819 * 0.6$, as illustrated in Fig. 7. From now on, we will focus on these ten movies throughout our running example. In real settings, the system would compute all the found movies.

In a second stage, the system processes the assertion of Quentin Tarantino as being a great director. As this informational element concerns a director (and not an individual of the class Movie), the system searches for films related to Quentin Tarantino through the isDirectorOf relationship. The system identifies the movies Kill Bill: Vol 1, Kill Bill: Vol 2, Pulp Fiction, Jackie Brown and Sin City and adds, for each one of them, a value of 1.0 to the Recon vector. The values of the Recon vector after this stage are shown in Fig. 8.

Finally, the system processes the information about Uma Thurman and, again, since it does not involve an evaluation about a movie, the system must search for the films that are related to this individual through the isActorIn relation. As Uma Thurman starred in Kill Bill: Vol 1, Kill Bill: Vol 2 and Pulp Fiction, the system will deduct 1.0 from these films' score in the Recon vector. The final values for each film in the Recon vector are shown in Fig. 9. Through this process, the movies that best fit the user's preferences are, in this example, Jackie Brown and Sin City.

3.3.2. Collaborative filter

In the previous section, the knowledge-based recommendation algorithm that the system uses is explained. This algorithm has been improved by including collaborative filtering techniques through which the social network can influence/affect the recommendations made to users.

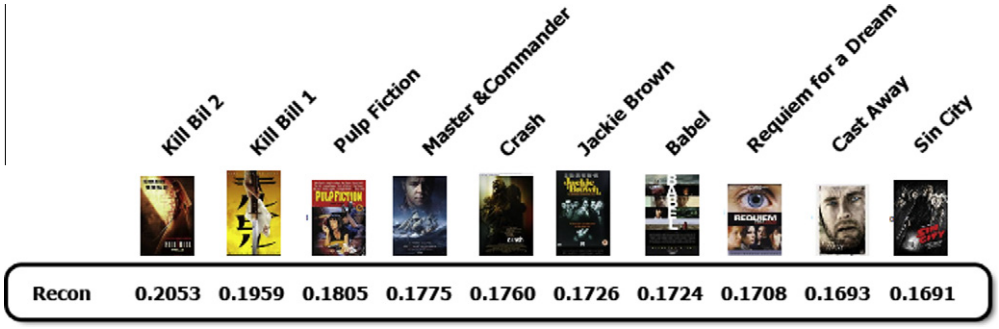

Fig. 7. The Recon vector after the analysis of Inglourious Basterds as being a Very Good movie.

Fig. 8. The Recon vector after the analysis of Quentin Tarantino as being a Great director.

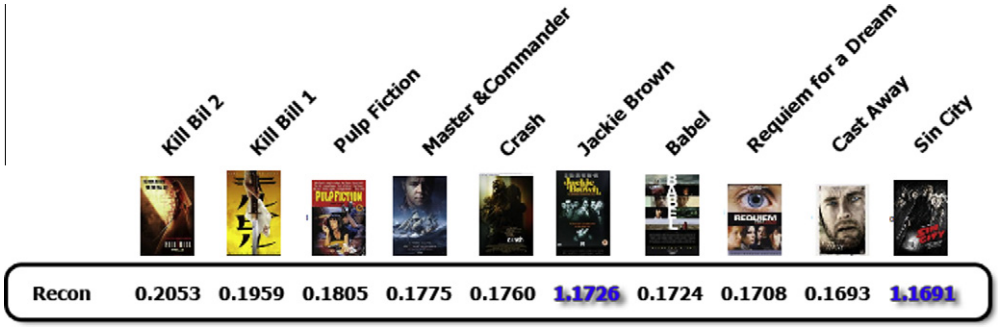

Fig. 9. Final vector Recon after classifying Uma Thurman.

Most collaborative approaches focus on finding users with similar interests in order to share recommendations among them. The approach proposed here is markedly different. It is based on how the social context can influence recommendations. For example, both FilmTrust (Golbeck & Hendler, 2006) and FOAFing-the-music (Celma, 2006) use the concept of trust by links of friendship through which the algorithm searches for recommendations. In this paper, the social network is handled differently. Our approach is based on the idea of the famous proverb, "Show me your friends and I'll tell you who you are". Similarly, it is reasonable to think that a person has similar interests to those of their friends. Thus, the proposed mechanism identifies the dominant content in the users' social network, that is, the content that has been better ranked by most of their friends. These preferences are taken into account to provide the recommendations in accordance with the users' chosen "social aperture". As it was pointed out before, some people have a more open mind to suggestions than others. The "social" receptivity set by users in their profile would describe the influence that the social network will have on their recommendations.

Users can select a "social aperture" profile on their personal preferences by choosing between three possible options: Conservative, Moderate and Liberal. The weight with which the system mixes the social network recommendations with the ones based on the users' knowledge base for the final recommendations vary according to the chosen option. A "Moderate" profile configures the system to weigh the recommendations based on the users' personal preferences with a 0.75 and the recommendations based on their social network with a 0.25 (see Fig. 10). A "Liberal" profile leads to the application of a 0.5 factor to both groups, and the "Conservative" profile does not allow the use of inherited recommendations through collaborative filtering.

To obtain the final recommendations it is necessary to calculate the list of recommendations based on the user preferences, as well as the list of the predominant recommendations among all users belonging to his/hers social network. The way the system calculates the recommendations based on the user's preferences is explained in the previous section.

Recommendations from the social network (i.e., collaborative filtering) are also based on the Recon vector. However, this time

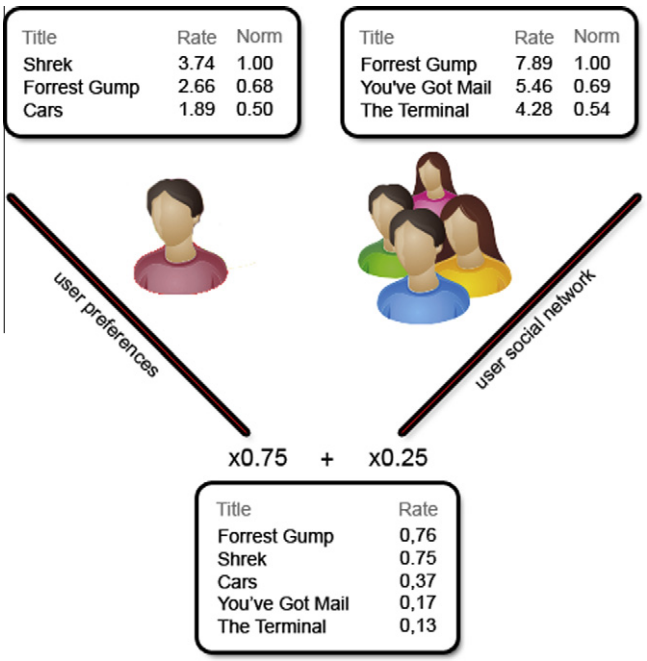

Fig. 10. Social influence.

it is necessary to consider all the vectors belonging to users that form part of the target user's social network. The collaborative filter is an average of the values of all the Recon vectors and, thus, represents the predominant recommendations and classifications made to and by the user's friends. It is reasonable to say that if a person has many friends who like the movie "A Clockwork Orange", is very likely that the user would also like that film, even though it had not been explicitly stated in his/her profile. Given that the interest rates are computed by the statements made by users, a movie can be rated on a much greater scale by simply receiving more "votes" than others. For instance, if a user states that he likes "Tom Hanks", the film "Cast Away" scores 1 point. If the user confirms that he likes the actress "Helen Hunt", the writer "William Broyles Jr.", the director "Robert Zemeckis", the genre "Drama" and the decade of the "2000s", the same film would add up a total of 6 points. Although collaborative filtering considers the average of these values, the data are subject to wide variations. This might lead to some undesired situations. For example, let us suppose that the movie with the highest rate (i.e., the one that obtains the highest value within the Recon vector) in the list of recommendations based on the user's preferences has a value of 2.5 and the one on the list of collaborative recommendations, adopting a moderate profile, obtained 7.6. In this case, the resulting rate for the movie recommended by the social network would weigh more than the one inferred by the personal preferences ($0.75 * 2.5 = 1.875 < 0.25 * 7.6 = 1.9$). In order to avoid this misled situation from happening, following the approach proposed in Claypool et al. (1999), the system normalizes the values of both lists on a scale of [0,1] and then weigh values according to the "social aperture" profile chosen by the user. Algorithm 4 (see Fig. 11) shows how to perform such normalization and add collaborative filtering to the final recommendation list.

The way this algorithm works is explained through another example. Let us suppose that the user A has a Recon vector containing the films (film [rate]) The Bone Collector [0.9], Rock [0.8] and The Italian Job [0.7]; user B has it with The Silence of the lambs [0.9], South Park [0.8] and The Italian Job [0.7] and user C with South Park [0.9], Panic Room [0.8] and The Italian Job [0.7]. Let us also suppose that they are all friends, which means that they share the same social network. When the recommender mechanism computes the recommendations to the user A, it separates

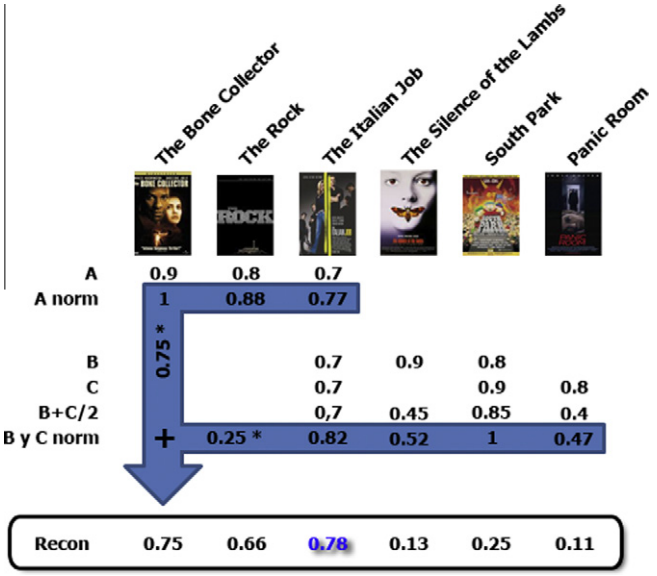

Fig. 12. Example of collaborative filtering using a "moderate" profile.

Algorithm 4: Collaborative filter

Input: Vector P containing recommendations based on the user's preferences ($P = recon_{user}$) where P_x represents the cumulative sum of a content x; Vector R containing recommendations based on the user's social network ($R = recon_{avgFriends}$) where R_x represents the average cumulative sum of a content x; Rate influence which represents the user's social aperture expressed in [0,0.5] (Conservative 0.0; Moderate 0.25; Liberal 0.5;);

Output: Sorted list Rank with final recommendations

```
index ← getKeys(P) ∪ getKeys(R)                    // movies from P and R
maxP ← max(P)                                      // highest value in P
maxR ← max(R)                                      // highest value in R
foreach x ∈ index do
    rateP ← P_x / maxP                             // normalizes P_x
    end
    rateR ← R_x / maxR                             // normalizes R_x
    Out[x] ← rateP * (1 − influence) + rateR * influence   // weights rateP and rateR
                                                   // according to the user's social aperture
    Rank ← rsortByValues(Out)                      // sorts movies index by values in a decrescent way
    return Rank
```

Fig. 11. Algorithm for calculating the collaborative filter.

all these data into two groups: one containing the values of Recon related to the profile of user A, and the other containing the average values from all the vectors that belong to users within the network of user A (i.e., average values from Recon of users B and C). Each set is normalized by its maximum value. In the case that user A has chosen a "moderate" social profile, the system will weigh the normalized values of these two groups with 0.75 and 0.25, respectively. Otherwise, if the user has a "Liberal" profile, the system uses equal weights for both groups, and if the user chose the "conservative" profile, the second group is simply discarded.

As shown in Fig. 12, thanks to collaborative filtering, the recommender system improves the classification of a film (The Italian Job) by inheriting evaluations from the network, thus increasing its importance in the final recommendation list.

4. Evaluation

This section focuses on the validation of the system developed in this work. To assess the quality of the recommendations made by the system, the metrics of precision, recall and F1-measure has been used. These evaluation measures are the ones usually applied to Information Retrieval systems (Díaz et al., 2003).

Precision measures the ability of the system to suggest content that is truly relevant for users. It can be calculated by the formula (4):

$$\text{Precision} = \frac{\text{Correctly recommended content}}{\text{Total recommended content}}, \tag{4}$$

where Correctly Recommended Content is the set of relevant recommendations made to the user and Total Recommended Content is the total number of recommendations made to the user.

Recall measures the system's ability to gather the relevant content to the specific user. It measures the amount of relevant content, or in other words, contents that fit the user's preferences, recovered by the system. It can be calculated by Formula (5):

$$\text{Recall} = \frac{\text{Correctly recommended content}}{\text{relevant content}}, \tag{5}$$

where Correctly Recommended Content is the set of relevant recommendations made to the user and Relevant Content is the set of movies that are interesting according to the user's likings.

F1-Measure is the harmonic mean between precision and recall. It corresponds to the Formula (6).

$$\text{F1-measure} = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}. \tag{6}$$

As the relevance of a recommendation is subjective to each person, it is not possible to automate the assessment process. Thus, to evaluate the system ten students were asked to interact with it. The students share the same social network, that is, there are bonds of friendship between them so that it is also possible to test the collaborative

User	Relevant				Irrelevant				Recommended			Precision			Recall			F-measure		
	Total	Correct			Total	Correct														
		C	M	L		C	M	L	C	M	L	C	M	L	C	M	L	C	M	L
A	21	20	20	21	19	5	4	3	32	35	37	59%	57%	57%	95%	95%	100%	73%	71%	72%
B	24	20	22	23	16	3	2	1	33	36	38	61%	61%	61%	83%	92%	96%	70%	73%	74%
C	21	17	21	21	19	13	7	2	23	33	38	74%	64%	55%	81%	100%	100%	77%	78%	71%
D	23	21	23	23	17	13	11	11	25	29	29	84%	79%	79%	91%	100%	100%	87%	88%	88%
E	22	19	20	20	18	7	5	5	29	33	34	65%	60%	59%	86%	90%	90%	74%	72%	71%
F	27	23	25	27	13	7	3	2	29	36	38	79%	70%	65%	85%	92%	100%	81%	79%	79%
G	18	17	18	18	22	14	8	4	23	29	35	73%	62%	51%	94%	100%	100%	82%	77%	68%
H	23	19	21	22	17	9	5	2	30	34	36	63%	62%	61%	82%	91%	96%	71%	74%	75%
I	22	20	20	21	19	7	5	3	32	35	37	62%	57%	57%	90%	90%	95%	73%	70%	71%
J	27	22	24	25	13	6	4	2	32	33	37	69%	73%	68%	81%	89%	93%	75%	80%	79%
	23	20	21	22	17	8	5	3	29	33	36	69%	64%	61%	87%	94%	97%	76%	76%	75%

Fig. 13. Precision and recall – assessment results.

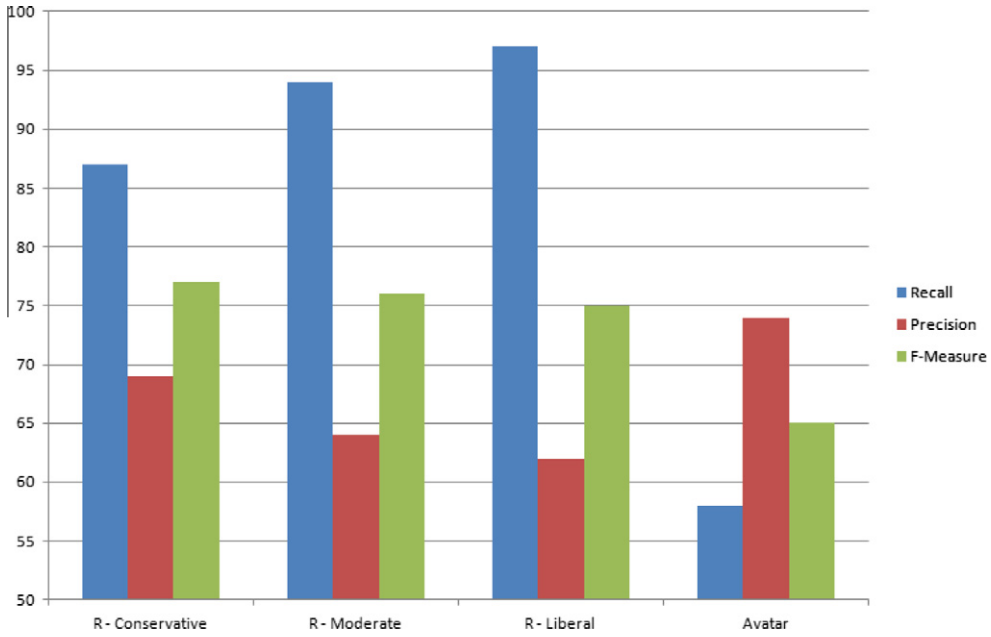

Fig. 14. Analyzing and comparing precision and recall.

filtering mechanism. The students set up their personal preferences by introducing a minimum of information in the system: choosing a minimum of 10 movies they liked, 10 that they did not like, actors and actresses who they like and dislike as well as genders they like/dislike. To determine the precision and recall of the system 40 films in the database were randomly selected and each participant in the experiment was requested to classify each one of these films between relevant and not relevant to their personal interests. "Relevant" movies represent "good" movies according to the users' preferences (regardless of whether they have seen them or not), and "not relevant" the contrary.

Once all films have been properly classified, three sets of recommendations are generated: one adopting the Conservative profile, another using the Moderate profile and, finally, on assuming the Liberal profile. Please keep in mind that the Conservative profile precludes the use of the collaborative filter, the Moderate profile weights recommendations from personal preferences and the social network with a factor of 0.75 and 0.25, respectively, and the Liberal profile mixes in equal proportion these suggestions. Afterwards, we verified, among these 40 films, how many of them were present in the lists of recommended items and how many recommendations were classified as "not relevant" by the user. The result of these measures can be seen in the table shown in Fig. 13.

Fields "Relevant - Total" and "Irrelevant - Total" represent the number of "good" and "bad" movies, respectively, classified by the user. "Relevant - Correct" and "Irrelevant - Correct" represent the hits of the system on identifying individuals in each group and the letters "C", "M" and "L" represent each of the aforementioned profiles: Conservative, Moderate and Liberal, respectively.

The results are quite promising. For example, in the best case the system was able to guess correctly 21 out of 25 recommendations made to a user, obtaining an impressive precision of 84% (user D). Its worst performance was 20 correct suggestions out of 32 (user A), maintaining a fairly high average precision of about of 70% in the 10 surveys. Besides the recall rates always hold more than 80% considering the Conservative profile. When adding social heritage to the recommendations, the quality of the recommendations tends to fall because more contents are recommended, which, on the other hand, improves the recall rates. Since there is no a significant representation of the social network available, this paper leaves a deeper analysis and evaluation of this feature as future work. In order to better analyse the performance of the proposed system, these results were compared with those obtained by the AVATAR system and published in Blanco-Fernández et al. (2008b). Fig. 14 presents a chart comparing these two systems. AVATAR does have a slightly better precision while the recall index of our approach is significantly greater, which ends up producing a better F-measure.

5. Conclusions and future work

Understanding and controlling recommender systems are very difficult tasks due to their complexity (Massa & Bhattacharjee, 2004). In this paper, a system for recommending content, products and services based on Semantic Web technologies and social networks is presented. This system tries to find similar content to users' preferences based on the semantics of such contents and the preferences of their social network. This system has been validated in the cinematographic domain with promising results.

Recommendation systems based on content are subject to problems such as overspecialization because they try to find content using their syntactic similarity. On the other hand, systems based on semantics or knowledge, such as AVATAR (Blanco-Fernández et al., 2008b), improve these results using Semantic Web technologies to identify contents semantically similar to the users' preferences, and try to diversify recommendations. Furthermore, in doing so, these systems also avoid the problem known as cold start, related to the provision of recommendations about new contents/products.

Collaborative filtering techniques have been used for a long time to generate groups of users with similar interests, but most of such techniques do not distinguish between friends and strangers. However, works such as Sinha and Swearingen (2001) show that users prefer recommendations from friends to those coming from third parties. Nowadays, with the mass adoption of social networks, this approach is viable and proposals as Liu and Lee (2010) focus on the use of these resources.

Another popular technique is collaborative tagging systems as used by Del.icio.us. In Liu, Zhang, and Zhou (2010), a way of addressing the cold start problem is presented. This is done through collaborative tags systems, which take a similar approach to the solutions based on the concept of semantic feedback, such as Garden and Dudek (2005).

This paper shows a hybrid system that put together concepts of knowledge and collaborative filtering that allows suggestions to be made based on (1) the semantic linkage between contents and users' preferences, and (2) also inherit recommendations from the users' social network. This work also introduce the concept of "social aperture", by means of which users can decide the impact social networks will have on the recommendations.

Refining and adjusting recommendation algorithms is always a challenge with many wide open possibilities. A possible and reasonable improvement of the proposed implementation could be not to limit the calculation of similarity only between individuals from a single main class, but to allow the calculation of similarity between two individuals belonging to two different classes. In our case, the main class is "Movie", but it would be very interesting to compare and recommend individuals of other classes such as actors or directors. For example, when a user likes "Chuck Norris", the system infers only on the films that this actor appeared in, but it would be interesting to also rate movies where similar actors have acted, such as "Steven Seagal". The main problem here is the complexity and the computing power necessary to carry out such calculations in real time. Using the technique proposed in Garden and Dudek (2005), in which users can semantically classify contents, could be an important step forward in improving the system's functionality and performance. Another crucial improvement is to implement a method to avoid "profiles attacks" as well as to evaluate the performance of our collaborative filtering using a more representative database. Adding more groups to the collaborative filtering service can also be advantageous. "Experts" can be treated differently from other groups such as "close friends" and "universal predominance". This might reflect better recommendations to the system as proposed in Amatriain et al. (2009).

Finally, the prototype has been validated in the movie domain. The next goal is to validate this system in a different one, such as the tourism domain, where user preferences can suggest various tourist and leisure activities.

Acknowledgments

This work is partially supported by the Spanish Government through Project SeCloud (TIN2010-18650).

References

Adams, J. M., Bennett, P. N., & Tomasic, A. (2007). *Combining Personalized Agents to Improve Content-Based Recommendations*. Pittsburgh: Language Technologies Institute, Carnegie Mellon University. CMU-LTI-07-015.

Adomavicius, G., & Tuzhilin, A. (2005). Towards the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering, 17*(6), 734–749.

Amatriain, X., Lathia, N., Pujol, J. M., Kwak, H. & Oliver, N. (2009). The wisdom of the few: a collaborative filtering approach based on expert opinions from the web. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. New York USA.

Arsene, O., Dumitrache, I., & Mihu, I. (2011). Medicine expert system dynamic Bayesian Network and ontology based. *Expert Systems with Applications, 38*(12), 15253–15261.

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web. *Scientific American Magazine, 284*, 34–43.

Blanco-Fernández, Y., Pazos-Arias, J. J., Gil-Solla, A., Ramos-Cabrer, M., López-Nores, M., García-Duque, J., et al. (2008a). A flexible semantic inference methodology to reason about user preferences in knowledge-based recommender systems. *Knowledge Based Systems, 21*(4), 305–320.

Blanco-Fernández, Y., Pazos-Arias, J. J., Gil-Solla, A., Ramos-Cabrer, M., López-Nores, M., García-Duque, J., et al. (2008b). An MHP framework to provide intelligent personalized recommendations about digital TV contents. *Software: Practice and Experience, 38*(9), 925–960.

Bogers, T. (2010). Movie recommendation using random walks over the contextual graph. In *CARS'10: Proceedings of the 2nd workshop on context-aware recommender systems*.

Burke, R. (2000). Knowledge-based recommender systems. *Encyclopedia of Library and Information Systems, 69*(Supplement 32), 175–186.

Celma, O. (2006). Foafing the music: Bridging the semantic gap in music recommendation. *The Semantic Web-ISWC, 2006*, 927–934.

Cheung, K. W., Kwok, J. T., Law, M. H., & Tsui, K. C. (2003). Mining customer product ratings for personalized marketing. *Decision Support Systems, 35*(2), 231–243.

Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D. and Sartin, M. (1999). Combining content-based and collaborative filters in an online newspaper. In *Proceedings of ACM SIGIR workshop on recommender systems. Citeseer*.

Dahlen, B. J., Konstan, J. A., Herlocker, J. L., Good, N., Borchers, A., & Riedl, J. (1998). *Jump-starting movielens: User benefits of starting a collaborative filtering system with dead data*. University of Minnesota TR, pp. 98-017.

Díaz, G. et al. (2003). La evaluación en recuperación de la información. *Hipertext. net, 1*. 7-0.

García-Manotas, I., Lupiani, E., Garcıa-Sánchez, F., & Valencia-García, R. (2010). Populating knowledge based decision support systems. *International Journal of Decision Support Systems Technology, 2*(1), 1–20.

García-Sánchez, F., Valencia-García, R., Martínez-Béjar, R., & Fernández-Breis, J. T. (2009). An ontology, intelligent agent-based framework for the provision of semantic web services. *Expert Systems with Applications, 36*(2), 3167–3187.

Garden, M. & Dudek, G. (2005). Semantic feedback for hybrid recommendations in Recommendz. In *EEE '05: Proceedings of the 2005 IEEE international conference on e-Technology, e-Commerce and e-Service (EEE'05)* (2005) (pp. 754–775).

Ghani R. & Fano A. (2002). Building recommender systems using a knowledge base of product semantics. In *Proceedings of the workshop on recommendation and personalization in ECommerce at the 2nd international conference on adaptive hypermedia and adaptive web based systems*. Citeseer.

Golbeck, J. & Hendler, J. (2006). Filmtrust: Movie recommendations using trust in web-based social networks. In *Proceedings of the IEEE Consumer communications and networking conference*, vol. 96. Citeseer.

Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM, 35*(12), 61–70.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition, 5*, 199–220.

Herlocker, J. L., Konstan, J. A., & Riedl, J. (2000). Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work* (pp. 241–250). ACM.

Jiang, X., & Tan, A. (2009). Learning and inferencing in user ontology for personalized semantic web search. *Information Sciences, 179*, 2794–2808.

Jung, J. J. (2011). Exploiting multi-agent platform for indirect alignment between multilingual ontologies: A case study on tourism business. *Expert Systems with Applications, 38*(5), 5774–5780.

Lam, X. N., Vu, T., Le, T. D., & Duong, A. D. (2008). Addressing cold-start problem in recommendation systems. In *Proceedings of the 2nd international conference on Ubiquitous information management and communication*, (pp. 208–211). ACM.

Liu, F., & Lee, H. J. (2010). Use of social network information to enhance collaborative filtering performance. *Expert Systems with Applications, 37*(7), 4772–4778.

Liu, Z. K. Z. C., Zhang, Y. C., & Zhou, T. (2010). Solving the cold-start problem in recommender systems with social tags. Arxiv, preprint, 2010.

Lupiani-Ruiz, E., García-Manotas, I., Valencia-García, R., García-Sánchez, F., Castellanos-Nieves, D., Fernández-Breis, J. T., et al. (2011). Financial news semantic search engine. *Expert Systems with Applications, 38*(12), 15565–15572.

Manber, U., Patel, A., & Robison, J. (2000). Experience with personalization of Yahoo! *Communications of the ACM, 43*(8), 35–39.

Massa, P. & Avesani, P. (2004). Trust-aware collaborative filtering for recommender systems. In *Proceedings of federated international conference on the move to meaningful Internet: CoopIS, DOA, ODBASE* (pp. 492–508).

Massa, P., & Bhattacharjee, B. (2004). Using trust in recommender systems: an experimental analysis. *Trust Management*, 221–235.

Mendenhall, W., & Sincich, T. (2006). *Statistics for engineering and the sciences*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994). Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work* (pp. 175–186). ACM.

Ruiz-Martínez, J. M., Miñarro-Giménez, J. A., Castellanos-Nieves, D., García-Sánchez, F., & Valencia-Garcia, R. (2011). Ontology Population: An application for the e-tourism domain. *International Journal of Innovative Computing, Information and Control (IJICIC), 7*(11), 6115–6134.

Ruiz-Martínez, J. M., Valencia-García, R., Fernández-Breis, J. T., García-Sánchez, F., & Martínez-Béjar, R. (2011). Ontology learning from biomedical natural language documents using UMLS. *Expert Systems with Applications, 38*(10), 12365–12378.

Shardanand, U. & Maes, P. (1995). Social information filtering: algorithms for automating word of mouth. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 210–217). ACM Press/Addison-Wesley Publishing Co.

Sinha, R. & Swearingen, K. (2001). Comparing recommendations made by online systems and friends. In *Proceedings of the DELOS-NSF workshop on personalization and recommender systems in digital libraries*. Citeseer.

Stanley, L., Lorenzi, F., Saldaña, R., & Licthnow, D. (2003). A tourism recommender system based on collaboration and text analysis. *Information Technology and Tourism, 6*(3), 157–165.

Studer, R., Benjamins, V. R., & Fensel, D. (1998). Knowledge engineering: Principles and methods. *Data and Knowledge Engineering, 25*, 161–197.

Swearingen, K., & Sinha, R. (2001). Beyond algorithms: An HCI perspective on recommender systems. In *ACM SIGIR 2001 workshop on recommender systems*. Citeseer, 2001.

Terveen, L., Hill, W., Amento, B., McDonald, D., & Creter, J. (1997). Phoaks: A system for sharing recommendations. *Communications of the ACM, 40*(3), 59–62.

Valencia-García, R., Fernández-Breis, J. T., Ruiz-Martínez, J. M., García-Sánchez, F., & Martínez-Béjar, R. (2008). A knowledge acquisition methodology to ontology construction for information retrieval from medical documents. *Expert Systems: The Knowledge Engineering Journal, 25*(3), 314–334.

Valencia-García, R., García-Sánchez, F., Castellanos-Nieves, D., & Fernández-Breis, J. T. (2011). OWLPath: an OWL ontology-guided query editor. *IEEE Transactions on Systems, Man, Cybernetics: Part A, 41*(1), 121–136.

Williams, C., Bhaumik, R., Sandvig, J. J., Mobasher, B., & Burke, R. (2008). *Evaluation of profile injection attacks in collaborative recommender systems*. Technical report. Available from http://facweb.cti.depaul.edu/research/techreports/TR06-006.pdf. (accessed 17.03.12).

Yang, H. (2009). Automatic generation of semantically enriched web pages by a text mining approach. *Expert Systems with Applications, 36*(8), 9709–9718.

Yang, S. Y. (2010). Developing an ontology-supported information integration and recommendation system for scholars. *Expert Systems with Applications, 37*(10), 7065–7079.

Yang, S. Y. (2011). OntoIAS: An ontology-supported information agent shell for ubiquitous services. *Expert Systems with Applications, 38*(6), 7803–7816.

Yildirim, H. & Krishnamoorthy, M. S. (2008) A random walk method for alleviating the sparsity problem in collaborative filtering. In *RecSys '08 Proceedings of the 2008 ACM conference on Recommender systems*.