Information Sciences 180 (2010) 4290–4311

Contents lists available at ScienceDirect

Information Sciences

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

A hybrid content-based and item-based collaborative filtering approach to recommend TV programs enhanced with singular value decomposition

Ana Belén Barragáns-Martínez [a,*], Enrique Costa-Montenegro [a], Juan C. Burguillo [a], Marta Rey-López [b], Fernando A. Mikic-Fonte [a], Ana Peleteiro [a]

[a] Telematic Engineering Department, University of Vigo, Vigo, Spain
[b] Consellería de Educación e O.U., Xunta de Galicia, Spain

ARTICLE INFO

Article history:
Received 21 August 2009
Received in revised form 13 July 2010
Accepted 22 July 2010

Keywords:
Recommender systems
Contents personalization
Collaborative filtering
SVD
Web 2.0

ABSTRACT

With the advent of new cable and satellite services, and the next generation of digital TV systems, people are faced with an unprecedented level of program choice. This often means that viewers receive much more information than they can actually manage, which may lead them to believe that they are missing programs that could likely interest them. In this context, TV program recommendation systems allow us to cope with this problem by automatically matching user's likes to TV programs and recommending the ones with higher user preference.

This paper describes the design, development, and startup of queveo.tv: a Web 2.0 TV program recommendation system. The proposed hybrid approach (which combines content-filtering techniques with those based on collaborative filtering) also provides all typical advantages of any social network, such as supporting communication among users as well as allowing users to add and tag contents, rate and comment the items, etc. To eliminate the most serious limitations of collaborative filtering, we have resorted to a well-known matrix factorization technique in the implementation of the item-based collaborative filtering algorithm, which has shown a good behavior in the TV domain. Every step in the development of this application was taken keeping always in mind the main goal: to simplify as much as possible the user task of selecting what program to watch on TV.

1. Introduction

The term *information overload* has become synonymous with the Internet and, specifically, with the problem of using the Internet to find the right information at the right time. The world of digital TV systems (together with cable and satellite services) shares this same luck because the number of channels and programs available to consumers is constantly increasing. This often means that viewers receive much more information than they can actually manage, which may lead them to believe that they are missing programs that could likely interest them. In this paper, we focus on this problem: it becomes almost impossible for people to cope with the choice of hundreds of TV channels and thousands of TV programs on a given day. We argue that even Electronic Program Guides (EPGs) no longer provide any real or practical assistance because,

* Corresponding author. Address: Departamento de Enxeñería Telemática, E.T.S.E. Telecomunicación, Vigo 36310, Spain. Tel.: +34 986 813473; fax: +34 986 812116.
E-mail address: belen@det.uvigo.es (A.B. Barragáns-Martínez).
URL: http://www.det.uvigo.es/~belen (A.B. Barragáns-Martínez).

0020-0255/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2010.07.024

although EPGs can increase the accessibility of TV programs, users tend to feel overwhelmed by the provision of too many program options.

In this context, researchers have begun to develop TV recommendation mechanisms to help users access the TV programs of their choice. Recommender systems provide suggestions for items likely to be of interest to a user. They usually operate by collecting viewing history data over a period of time, gleaning user viewing preferences from the data, mapping user preferences for TV program attributes, filtering non-interesting programs, and finally recommending an appropriate program [9,4].

Recommendation techniques come in two basic flavors: *collaborative filtering* and *content-based filtering*. *Collaborative filtering* (CF) approaches [40] rely on the availability of user ratings information and make suggestions to a target user based on the items that similar users have liked in the past, without relying on any information about the items themselves other than their ratings. CF models the social process of asking a friend for a recommendation, making suggestions for some target user from the items that are liked by users similar to him/her (user-based) or from items that have received similar ratings to the items that the target user likes (item-based).

In contrast, *content-based filtering* techniques [33] rely on item descriptions and generate recommendations from items that are similar to those the target user has liked in the past, without directly relying on the preferences of other users.

Although both types of recommenders can assist TV viewers with program selection, they both have some weaknesses. The primary drawback of content-based filtering systems is their tendency to *over-specialize* the item selection and only very similar items to previous items consumed by the user are recommended. User studies have shown that users find online recommenders more useful when they recommend unexpected items [30].

There are problems too with the standard CF approach. The so-called *grey-sheep* problem causes difficulties when a particular user's ratings history does not help the system identify a set of similar recommendation partners. Similarly, the *cold-start* problem is caused by new users in the system which have not submitted any ratings. Thus, the system is not able to infer the user's preferences and generate recommendations until a few items have been rated. In addition, for a CF system to work well, several users must evaluate each item; even then, new items cannot be recommended until some users have taken the time to evaluate them (*first-rater* problem). The system is therefore unable to generate semantic interconnections to these items, and therefore they are never recommended. Moreover, it is common for the user-item ratings matrix to be *sparsely* populated, making it difficult again to confidently identify similar users and items, due to a lack of ratings overlap. Another important limitation of CF systems is their lack of *scalability*: as the number of users and items increases, this approach may lead to unacceptable latency for providing recommendations during user interaction.

The first three problems of CF systems can be solved by using content-based techniques because they allow the user to encounter new content that has not been rated before, since the system must be capable of matching the characteristics of an item against relevant features in the user's profile. In the same way, they permit the new user to find interesting contents. Moreover, the combination of CF and content-based techniques allows the elimination of the *over-specialization* problem of pure content-based recommenders. Therefore, to overcome all these limitations, we propose to adopt a hybrid approach between content-based matching and collaborative filtering. This hybrid proposal is specially interesting in the domain of TV recommenders, since TV programs recommendation has been typically carried out through personalization of the Electronic Program Guide (EPG), using only content-based filtering techniques. At the initial stage of use, program recommendation can be inferred from the viewer's preferences when the information about the user's viewing history is not available yet. As the system gathers more information about the user's viewing history and the given ratings, it can fine-tune program recommendations to match the personal preferences of an individual user using the CF approach.

But there are still some other limitations, mentioned above but not yet solved, of collaborative filtering. Although this method has been shown to produce high quality recommendations, its performance degrades with the increasing number of users and programs (*scalability* problem). Aside from that, the low density of ratings, also known as *sparsity* problem (that is, each user rates only a small fraction of the total amount of available programs), causes bad quality predictions. Therefore, new recommender system technologies are needed to quickly produce high quality recommendations, even for very large-scale problems.

To solve both problems, we have explored the technology called *singular value decomposition* (SVD) to reduce the dimensionality of the recommender system database. Particularly, we have adapted the solution proposed by Vozalis and Margaritis [55]. SVD comes from the area of linear algebra and has been successfully employed in the domain of information retrieval. Recently, various recommender systems researchers have suggested its use as a possible solution to the above mentioned problems.

Taking into account all the issues mentioned above, we have also resorted to social collaboration to develop **queveo.tv** [39], an innovative web 2.0 application (which has been overviewed in [6]) that learns about the TV viewing preferences of individual users, so as to provide them with highly customized daily TV content recommendations. In short, each user receives a list of programs to watch today, tomorrow, or the day after tomorrow, which has been made with his/her[1] preferences in mind, listing only those programs that she will likely be interested in watching (or recording).

In the literature, we can find many hybrid systems but, to the best of our knowledge, no previous proposal addresses all our goals and efficiently solves all the problems of both content-based and CF filtering. We have combined the most used

[1] For the sake of simplicity, hereinafter, we will use either "her" or "she" when "his/her" or "he/she" should be, respectively, used.

technique by the recommenders in the TV domain, i.e., content-based filtering, with the most well-known and successful recommender method, i.e., the collaborative filtering, in its more accurate version (the item-based approach). To demonstrate the feasibility of the proposed approach, we will evaluate the algorithms in terms of accuracy, and we will show how queveo.tv was implemented as a real application which provides all typical features of any social network, such as supporting communication among users (via private messaging) as well as allowing users to add and tag contents, rate and comment the items, among other functionalities.

The rest of this paper is organized as follows. Section 2 discusses previous related work, compares content-based and collaborative filtering techniques in the TV domain, and provides an analysis of the factors which make CF techniques to work well in this domain. Section 3 describes the architecture of the system, and details every design decision. Section 4 describes the content-based filtering and collaborative filtering strategies. The details of the SVD technique are provided in Section 5 together with our version of the CF algorithm enhanced with SVD. In Section 6, we present the evaluation results, whereas the prototype implementation and its launch are described in Section 7. Finally, Section 8 concludes and points out directions for future work.

2. Background

Recommender systems suggest items of interest to users based on their explicit and implicit preferences, the preferences of other users, and user and item attributes. Thus, based on how recommendations are made, recommenders systems are usually classified into the following two categories [1,2,5]:

Content-based recommendations: Content-based recommendation systems recommend an item to a user based upon a description of the item and a profile of the user's interests. Content-based recommendation systems may be used in a variety of domains ranging from recommending web pages, news articles, restaurants, television programs, and items for sale. Although the details of various systems differ, content-based recommendation systems share in common a means for: (i) describing the items that may be recommended; (ii) creating a profile of the user that describes the types of items the user likes; and (iii) comparing items to the user profile to determine what to recommend [33]. Notice that recommendations are made without relying on information provided by other customers, but solely on items' contents and users' profiles. In content-based filtering, the features used to describe the content are of primary importance. The more descriptive they are, the more accurate the prediction is.

Collaborative filtering: These techniques generally involve matching the ratings of the current user for objects (e.g., movies or products) with those of similar users (nearest neighbors) in order to produce recommendations for objects not yet rated or seen by the active user. *User-based* and *item-based* collaborative filtering are two basic variants of this approach. Within the former category, traditionally, the primary technique used to accomplish this task is the standard memory-based *k*-Nearest-Neighbor (*k*NN) classification approach which compares a target user's profile with the historical profiles of other users in order to find the top *k* users who have similar tastes or interests [30].

According to [10], algorithms for collaborative recommendation can be grouped into two classes: *memory-based* and *model-based*. Memory-based algorithms [10,40] are heuristics that make ratings predictions based on the entire collection of items previously rated by users. These systems require all ratings, items, and users to be stored in memory. Model-based algorithms [15] use the collection of ratings to learn a model, which is then used to make ratings prediction. These systems periodically create a summary of ratings patterns offline. Therefore, in comparison with model-based methods, memory-based algorithms can be considered "lazy learning" methods because they do not build a model, instead they perform the heuristic computations when recommendations are requested [1]. The advantage of memory-based methods over their model-based alternatives is that they have less parameters to be tuned, while the disadvantage is that the approach cannot deal with data sparsity in a principled manner. Pure memory-based models do not scale well for real-world application. Thus, almost all practical algorithms use some form of pre-computation to reduce run-time complexity. As a result, current practical algorithms are either pure model-based algorithms or a hybrid of some pre-computation combined with some ratings data in memory.

The most critical issue with memory-based algorithms is how to determine the similarity between two users. The two most popular approaches are the *correlation-based approach* [40] and the *cosine-based approach* [10,41].

Although the classification of CF techniques in memory-based and model-based is the most common one, authors in [43] explore a different organization of collaborative filtering algorithms: *non-probabilistic algorithms* and *probabilistic algorithms*. They consider algorithms to be probabilistic if they are based on an underlying probabilistic model. That is, they represent probability distributions when computing predicted ratings or ranked recommendation lists. In general, non-probabilistic models are widely used by practitioners. However, probabilistic models have been gaining favor, in particular in the machine learning community.

2.1. Comparing collaborative filtering to content-based filtering in the TV domain

Collaborative filtering uses the assumption that people with similar tastes will rate the TV programs similarly. Content-based filtering uses the assumption that TV programs with similar features will be rated similarly.

Table 1
Content-based filtering vs. collaborative filtering.

	Content-based recommendation	Collaborative filtering
Strength	No need of ratings avoiding any *cold-start* problem Resource consumption for computation is low High coverage: The range of items available for recommendation is wide	Diversity: It can deal with any kind of content Serendipity: It provides items with dissimilar content with those experienced in the past No need of user profiles nor contents metadata
Weakness	Need of content to analyze Over-specialization: The user is restricted to seeing items similar to those already experienced	Sparsity: The lack of user preference data causes a performance decline, and makes it difficult to find nearest-neighbors for users with peculiar taste Scalability: Increase of user preference data leads to a performance improvement, but much more resources are consumed *Grey sheep*, *first-rater*, and *cold-start* problems

Content-based filtering and collaborative filtering have long been viewed as complementary [2]. Table 1 shows a comparison between both techniques, highlighting their main advantages and disadvantages, some of which are discussed in the following paragraphs.

On the one hand, content-based filtering can predict relevance for programs without ratings (e.g., new shows or hardly seen programs) whereas collaborative filtering needs ratings for a program in order to predict for it. On the other hand, content-based filtering needs content to analyze. Sometimes content is either scarce or incorrect (e.g., some programs can be badly tagged), which is very negative because the system depends on the availability of content descriptions of the items being recommended. Collaborative filtering does not require content. A content filtering model can only be as complex as the content to which it has access. For instance, if the system only has genre metadata for movies, the model can only incorporate this one extremely coarse dimension. Furthermore, if there is no easy way to automatically extract a feature, then content-based filtering cannot consider that feature. Collaborative filtering allows evaluation of such features, because people are doing the evaluation.

As mentioned above, content-based filtering may over-specialize. The programs which are recommended are the ones that match the content features in the user's interest profile. Hence, those programs which do not contain the exact features specified in the interest profile may not be recommended even if they are similar (e.g., due to synonymy in keyword terms or the bad tagging of programs). Researchers generally believe that collaborative filtering leads to more unexpected or different programs that are equally valuable. Some people call this property of recommendations novelty or serendipity [21]. However, collaborative filtering has also been shown to over-specialize in some cases [62]. With respect to CF disadvantages, we list in Table 1 those already discussed in Section 1. In spite of the apparent high number of disadvantages of the CF approach, we have made use of it taking into account the thorough analysis provided in Section 2.3 focused on the TV domain.

2.2. Hybrid approaches

In order to exploit the advantages of available recommendation methods, several hybrid approaches have been proposed, in their vast majority concerning combinations of collaborative filtering and content-based filtering. A good review of hybrid approaches can be found in [12,2].

Burke [12] classifies hybridization techniques into seven classes:

1. *weighted*, where each of the recommendation approaches makes predictions which are then combined into a single prediction;
2. *switching*, where one of the recommendation techniques is selected to make the prediction when certain criteria are met;
3. *mixed*, in which predictions from each of the recommendation techniques are presented to the user;
4. *feature combination*, where a single prediction algorithm is provided with features from different recommendation techniques;
5. *cascade*, where the output from one recommendation technique is refined by another;
6. *feature augmentation*, where the output of one recommendation technique is fed to another; and
7. *meta-level*, in which the entire model produced by one recommendation technique is utilized by another.

Switching, mixed, and weighted hybrids are differentiated from the remaining techniques in Burke's taxonomy by the fact that each of the individual (base) recommendation methods produces a prediction, independently from each other, which is then presented to the user either as a single combined prediction (switching, weighted) or as two independent predictions (mixed). Switching hybrids, in particular, are low-complexity hybridization methods based on the examination of the conditions that affect the performance of the base algorithms each time a prediction is requested. When certain conditions occur, the final prediction is the outcome of the base recommendation approach that is not affected (or is less affected) from these conditions.

Our proposal makes use of a mixed hybridization technique which offers not only content-based and collaborative filtering recommendations, but also a list of *star recommendations*, composed of those shows which appear in the former lists.

They are presented to the user after being sorted according to the averaged rating given by all users in the system to this show in question.

2.3. Why would CF techniques work well in the TV domain?

As explained in [43], CF is better known to be effective in domains with certain properties. It seems useful to acquaint ourselves with them, and consider whether the devised application is a good fit. Schafer et al. [43] group these properties below into *data distribution*, *underlying meaning*, and *data persistence*. In the following paragraphs, we will simply list them to check if collaborative filtering can work well in the TV domain.

2.3.1. Data distribution

1. *There are many items.* If there are few items to choose from, the user can learn about them all without need for computer support. It is clear that having (currently in queveo.tv) more than 9, 367 TV programs and 340, 181 broadcastings in 133 channels we need a recommender system to find the most interesting programs.
2. *There are many ratings per item.* If there are few ratings per item, there may not be enough information to provide useful predictions or recommendations. This will depend on the popularity level of each TV program, because shows which are aired in popular channels at prime time will doubtless accumulate the biggest amount of ratings (either positive or negative) whereas programs broadcast through local channels with small audience will hardly have ratings. This way, ratings distribution is very skewed: a few items get most of the ratings whereas a long tail of items get few ratings. Items in this long tail will not be confidently predictable. But this latter kind of programs can also be recommended using the content-based filtering.
3. *There are more users rating than items to be recommended.* A corollary of the previous paragraph is that often you will need more users than the number of items that you want to be able to capably recommend. More precisely, if there are few ratings per user, you will need many users. Lots of systems are like this. Currently, queveo.tv has more items than users (almost one thousand users), but it can be supposed that in a long term basis there will be more users than programs.
4. *Users rate multiple items.* If a user rates only a single item, this provides some information for summary statistics, but no information for relating the items to each other. In the TV domain, it is common that users rate many programs even at the same time, because in the same session they use to rate all programs recently seen.

2.3.2. Underlying meaning

1. *For each user of the community, there are other users with common needs or tastes.* CF works because people have needs or tastes in common. If a person has tastes so unique that they are not shared by anybody else, then CF cannot provide any value. More generally, CF works better when each user can find many other users who share their tastes in some fashion. In the TV domain, it is clear that users share tastes: they like the same kind of series, or realities, etc. In fact, they group themselves to share their opinions about particular TV programs: queveo.tv has currently 43 groups specialized in trendy series, soccer, F1, magazines, etc.
2. *Item evaluation requires personal taste.* In cases where there are objective criteria for goodness that can be automatically computed, those criteria may be better applied by other means than collaborative filtering, e.g., search algorithms. Collaborative filtering allows users with similar tastes to inform each other. CF adds substantial value when evaluation of items is largely subjective, or when those items have many different objective criteria that need to be subjectively weighted against each other. Sometimes there are objective criteria that can help, but if recommendation can be performed using only objective criteria, then CF is not useful. In the TV domain, part of the recommendation can be done objectively, making use of the user profile, but CF is necessary to help a user discover programs not ever seen, that may belong to genres not included in her favorite ones but probably liked by the user.
3. *Items are homogeneous.* That is to say, by all objective consumption criteria they are similar, and they differ only in subjective criteria. Music albums are like this. Most are similarly priced, similar to buy, of a similar length. This characteristic can result essential in commercial systems, but it is not our case, although we can consider that all items are homogeneous.

2.3.3. Data persistence

1. *Items persist.* A CF system does not only need a single item to be rated by many people, but it also requires that people share multiple rated items, i.e., that there is an overlap in the items they rate. Considering the domain of TV programs, in queveo.tv new broadcastings of programs appear each day, these broadcastings can be rated, and the average of these ratings is the program rating. When recommending broadcastings to a user, they are presented once sorted according to the program rating (not the broadcasting rating). Thus, it is not needed that the current broadcasting has been previously rated but any other previous broadcasting of the same program. In general terms, programs are scheduled daily or weekly so we can consider that items persist. In order for a CF system to generate a prediction for user u_a regarding a recently appeared broadcasting, a typical CF algorithm requires that: (i) one or more users have rated the TV program

(any previous broadcasting); and (ii) these users have also rated some other programs which user u_a has also rated. In our domain, all of this means that although broadcastings are only important for a short time, programs persist, so CF is useful and necessary.

2. *Taste persists.* CF has been most successful in domains where users' tastes do not change rapidly (e.g., movies, books, and consumer electronics). If tastes change frequently, then older ratings may be less useful. In the TV domain, taste generally persists.

As a conclusion, we have that seven of the above nine properties are fully satisfied in the TV domain, which leads us to believe that CF fits very well with our purposes.

2.4. Previous research in TV program recommendation systems

The first attempt to face the problem of TV content overload was done by most of the systems through the adoption of EPGs (Electronic Program Guides) [58]. These guides provide an on-screen menu system to help users navigate the TV listings maze. However, EPGs do not scale well and viewers do not have patience to surf on so many pages with programs of hundreds of channels. Thus, the personalized TV recommendation systems emerge.

Among the initial proposals, we can distinguish between two main categories:

(i) those works built on top of set-top-boxes, as for example P-EPG [14] and Multi-Agent [24]; and
(ii) those web-based, for instance, PTV [44,45], TV-Scout [8,9] and the work in [60].

Those falling in the first category focus on improving basic functions of EPGs as program navigation, as well as building the user profile for each viewer to recommend TV programs that best match the individual taste. They construct an accurate user profile by automatically tracking each user's TV viewing patterns. Thus, their main advantage is that this implicit profile is made automatically, requiring no initialization from the user. The more limiting drawback of this kind of systems is that they cannot use other users' preferences to create recommendations such as those created by collaborative filtering. The reason is that they cannot exchange the preferences, and therefore they all are content-based only.

This problem does not appear within the web-based recommenders, where we should highlight PTV, one of the first hybrid proposals between content-based and collaborative filtering techniques, and maybe the most complete work in the initial literature. In contrast to our proposal, users are invited to provide lists of program titles: both a positive list containing programs that the user has liked in the past, and a negative list containing programs that the user has disliked. We believe this is against one of the basic principles of recommenders, that is, the discovery of new programs, in such a way that the user cannot classify all programs in likes or dislikes because she may not even know their existence. Another difference is that, instead of using item-based CF (as queveo.tv does), PTV implements user-based CF. There is evidence that item-based nearest neighbor algorithms are more accurate in predicting ratings than their user-based counterparts [42]. Additionally, it does not present any solution to its latency problems.

In turn, other web-based system as TV-Scout [8,9] can be seen more as a retrieval system (with a complex filtering schema) than a recommender one. The users must specify a query to initiate the recommendation process. It seems to also use a hybrid approach between CF and content-based filtering techniques but, to the best of our knowledge, it is not clear what collaborative algorithm was implemented and how the sparsity and scalability problems are solved.

More recent works in this area are [61,22,53,11]. Authors in [61] present the architecture of a personalized TV system on the basis of TV-Anytime[2] metadata. As so many other works in this field, its main limitation is that they cannot consider other users' preferences to generate the recommendations.

AIMED [22] proposes also a hybrid recommendation mechanism whose content-based filtering technique considers tendencies like: mood, viewing behavior, demographic information, etc.; in order to obtain more accurate results. Its main weakness is that the collaborative counterpart is carried out by using a simple matching between an individual user and a viewer group model (obtained with the *K*-means method from the demographic, lifestyle, and program categories data). This matching allows the system to recommend a set of candidate programs which are preferred by the viewing group, which share a similar profile.

Similar to our content-based recommender, the *vector space model* is also used in [53]. In this interesting work, the *vector space model* is also used in a novel ICF (Indirect CF) algorithm, where the user preferences vector is replaced by the predicted preferences vector, which is obtained by making use of a user-based CF algorithm. Since this work was not evaluated with a standard metric, we are unable to compare their results with ours.

Finally, we also highlight the work in [11] where another hybrid method is shown, which combines content-based recommendation and *folksonomies* (collaborative and social recommendations). They use as a collaborative recommender a group of algorithms called Slope One [27], which is possibly the simplest method of non-trivial item-based collaborative filtering based on points, but they provide no solutions neither to the sparsity nor the scalability problems of CF algorithms.

[2] http://www.tv-anytime.org.

Summarizing, our proposal is the one, to the best of our knowledge, which effectively combines content-based with collaborative techniques, providing efficient solutions to the problems of both algorithms. Besides, taking advantage of the so-called Web 2.0, queveo.tv provides all typical advantages of any social network, such as supporting communication among users (via private messaging) as well as allowing users to add and tag contents, rate and comment the items, among other functionalities. This way, queveo.tv provides an ad hoc social community: the perfect meeting point where, e.g., everybody watching the same show on TV can comment, rate, tag, and definitely, enjoy themselves.

3. Architecture of the system

As seen in previous sections, in the TV domain, content-based filtering and collaborative filtering complement each other really well. Our system implements both techniques, as well as a merging algorithm that combines the results provided by both of them, which is done taking into account the averaged rating for each program. The CF algorithm can be viewed as a generalized CF approach which utilizes SVD as an augmenting technique in order to enhance the efficiency of the recommender system and improve the accuracy of the generated predictions. Our system uses *memory-based item-to-item correlation* to predict the level of interest for each item, and calculate the items' neighborhood. Our approach also makes use of *memory-based user-to-user correlation*, as the user neighborhood for a new user has to be computed during runtime of the system. In Fig. 1 we present an overview of our recommender framework.

The hybrid proposal works well because both algorithms complement each other in such a way that the content-based algorithm recommends the usual programs and CF provides the discovery of new shows. As explained in Section 2.2, when a new program appears in both lists of recommendations, it appears clearly highlighted in queveo.tv as a *star recommendation*.

3.1. Downloading TV programming information

Firstly, as we can see in Fig. 1, the system gathers all TV programming information of all the available channels (this number can be dynamically augmented), where each program is conveniently tagged with suitable keywords describing its content. Each listing entry in this database includes details such as the program name, the viewing channel, the start and end time, the associated tags, and typically some text describing the program in question. To store these entries, we use XMLTV format (which will be described more detailedly in Section 7). The schedule database is automatically constructed and updated on a daily basis by executing a parser which explores different websites with listings information. An example of a program description in queveo.tv can be seen in Fig. 2, which shows a screenshot of our website (in Spanish). Fig. 3 shows the TV listings scheduled for three days for one of the available channels.

3.2. Profile acquisition

All those TV listings (as we can see in Fig. 3) represent a great amount of information which should be filtered according to the user's preferences, for example, what kind of programs the user likes, what channels the user has access to, etc. In order to do this, the system must first construct a sufficiently-detailed model of the user's tastes through preference elicitation.

This process can be done either *explicitly* (by querying the user) or *implicitly* (by observing the user's behavior). Aside from asking for information about tastes when building the profile, demographic and lifestyle information (age, gender, profession, her TV schedule or leisure time) is also requested, as suggested in [26], obtaining a very complete profile. Both user profile and TV guide will be the input to the first filtering. Explicit and implicit data acquisition methods present advantages and disadvantages: implicit methods are considered unobtrusive to user's main goal in using a system, but explicitly acquired data are more accurate in expressing preferences or interests.

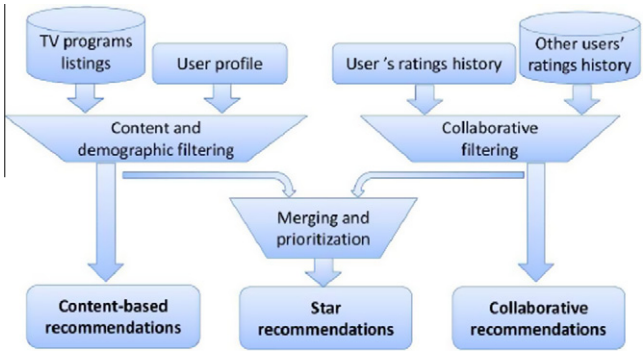

Fig. 1. Overview of our recommender framework.

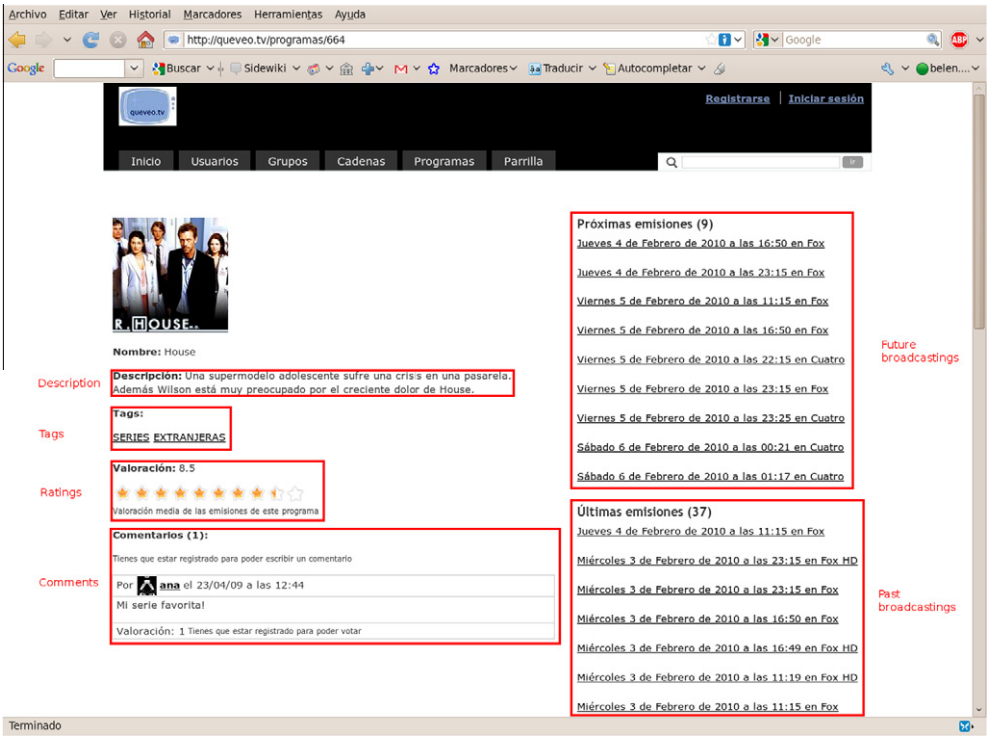

Fig. 2. Information about the TV series House showing its description, tags, rating, comments, and past and future broadcastings (right column) (queveo.tv snapshot).

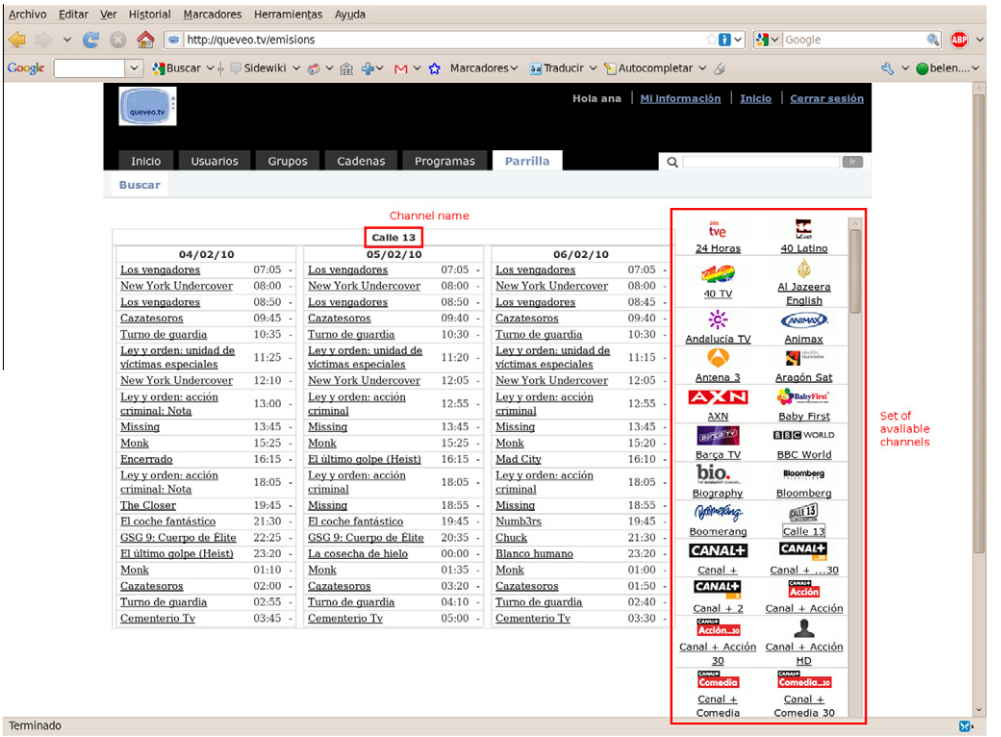

Fig. 3. TV shows scheduled for channel "Calle 13" for today, and the two following days (queveo.tv snapshot).

A profile of the user's interests is used by most recommendation systems. This profile may consist of a number of different types of information. In queveo.tv we concentrate on two types of information:

1. A model of the user's preferences, i.e., a description of the types of items which are interesting (or not) for the user, as well as demographic information, her TV schedule, etc.
2. A history of the user's interactions with the recommendation system. This includes the ratings given by the user to any program, previously recommended or not. Some other types of history include her recent activity, her groups, friends, etc.

3.2.1. Acquiring user preferences

In queveo.tv each model of user's preferences contains two types of information: domain preferences and program preferences. The former ones describe general user's preferences such as a list of available TV channels, preferred viewing times together with some demographic data. Program preferences are represented as two list of subject keywords or tags with her genre preferences. To collect this information, our recommendation system provides an interface that allows users to construct a representation of their own interests. Check boxes are used to permit a user to select from the known values of program genres, e.g., series, sports (soccer, basketball, tennis, motorcycling and formula 1), documentaries, news, movies, etc., the ones the user likes and hates. This information of the user is stored in the profiles database and used by the content-based recommendation engine whenever the user requests program recommendations. Figs. 4 and 5 show how user Ana fills out the data needed to know her preferences: her preferred schedule, her likes/hates, her favorite channels, etc.

We have observed that there is a main limitation of user customization systems: they do not provide a way to determine the order in which to present items, and can find either too few or too many matching items to display (as it is also pointed out in [33]). This problem does not arise in queveo.tv because selected programs are always sorted according to the averaged rating given by all users to that program. The first 10 results found of both kinds of recommendations are presented (if any) to the user who can then navigate to see the remainder of them.

To guarantee the smooth running of the queveo.tv's content-based recommender, the system must keep an accurate database of user profiles. Each profile encodes the TV preferences of a given user, a listing with channel information, preferred viewing times, program genres' likes and dislikes, etc. This profile information is specially needed to bootstrap the recommendation process when there is no rating information. An example of the appearance of a user profile in queveo.tv can be seen in Figs. 6 and 7.

3.2.2. Ratings gathering

Although each user's profile information is very important, specially at initial stages, the majority of information is learned from grading feedback provided by the user (each user can rate any program at any time). In queveo.tv, users are

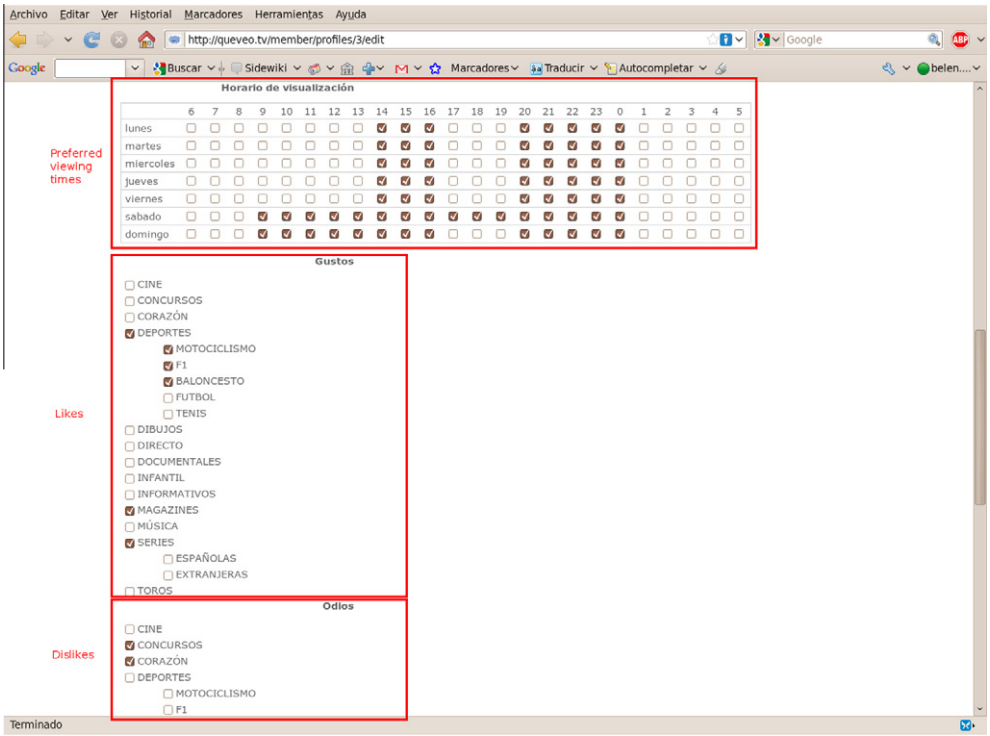

Fig. 4. Part of the questionnaire to be filled out by Ana where her preferred schedule, likes, and dislikes are requested (queveo.tv snapshot).

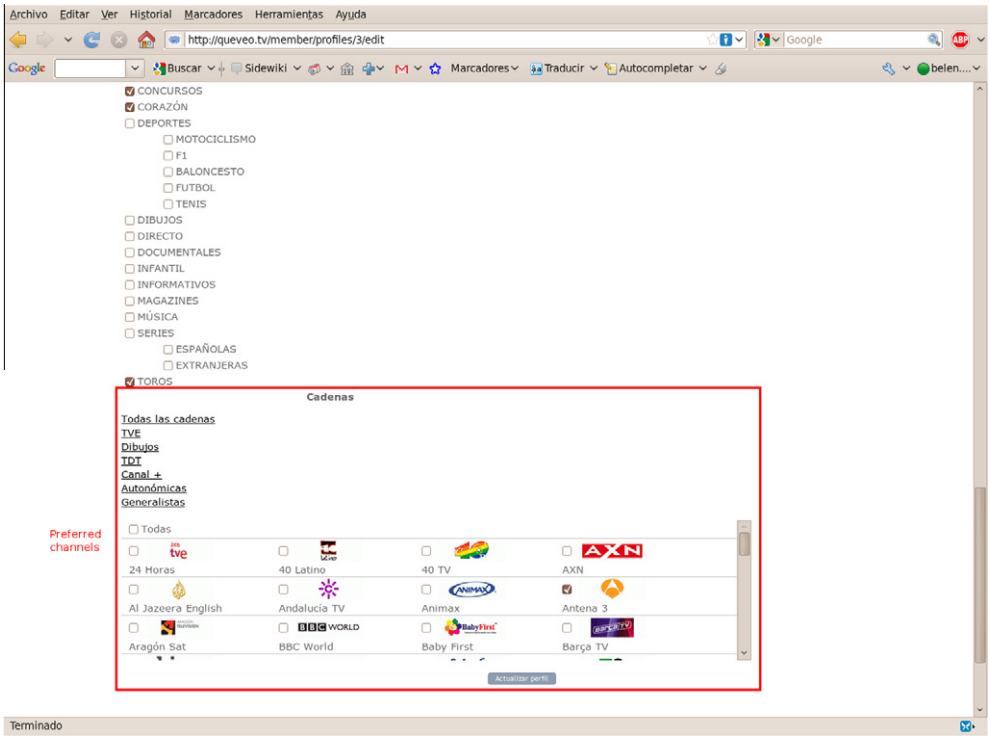

Fig. 5. Second part of profile questionnaire asking for Ana's preferred channels (queveo.tv snapshot).

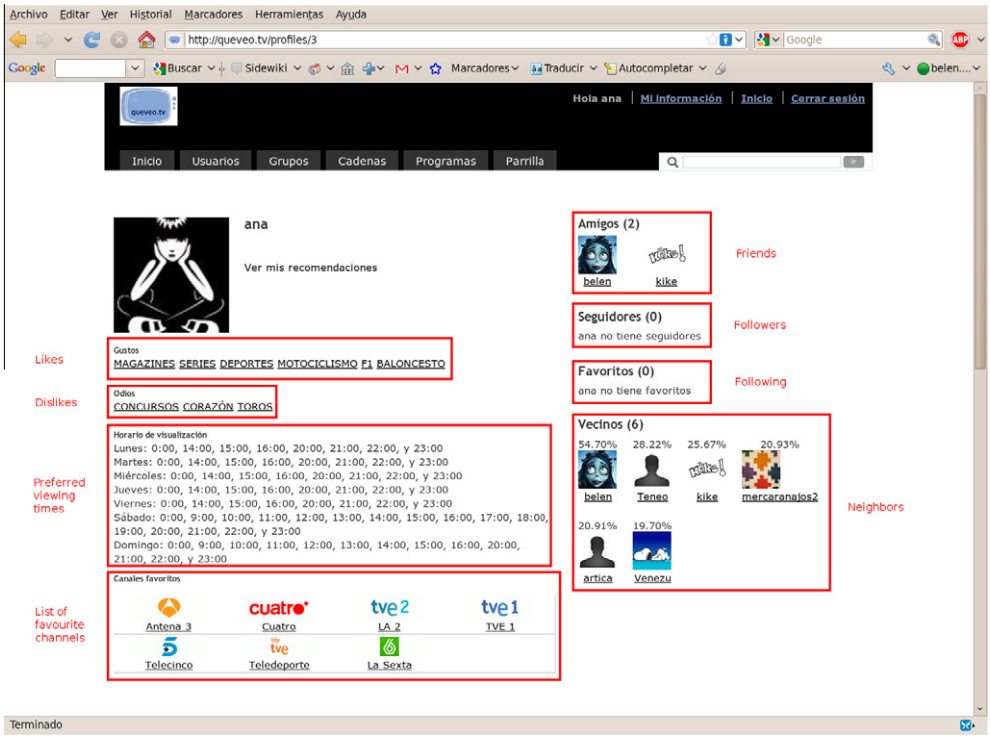

Fig. 6. Profile of user Ana showing her likes, dislikes, preferred schedule, favorite channels, and, in the right column, her friends, followers, favorite users, and neighbors (queveo.tv snapshot).

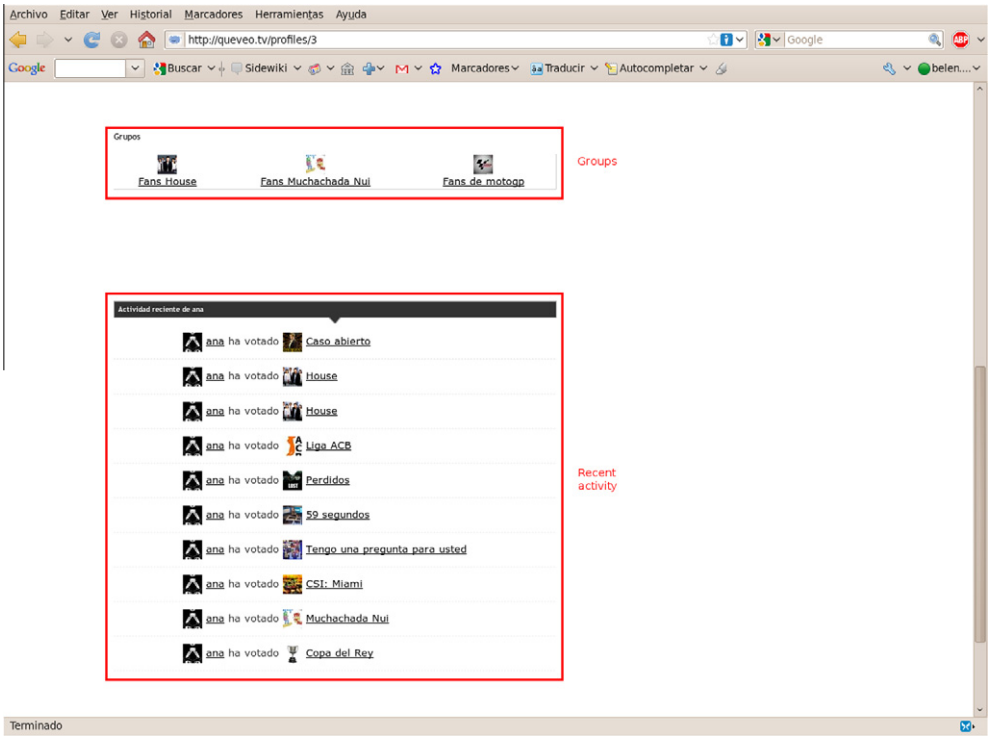

Fig. 7. Second part of Ana's profile showing her groups and summary of her recent activity (queveo.tv snapshot).

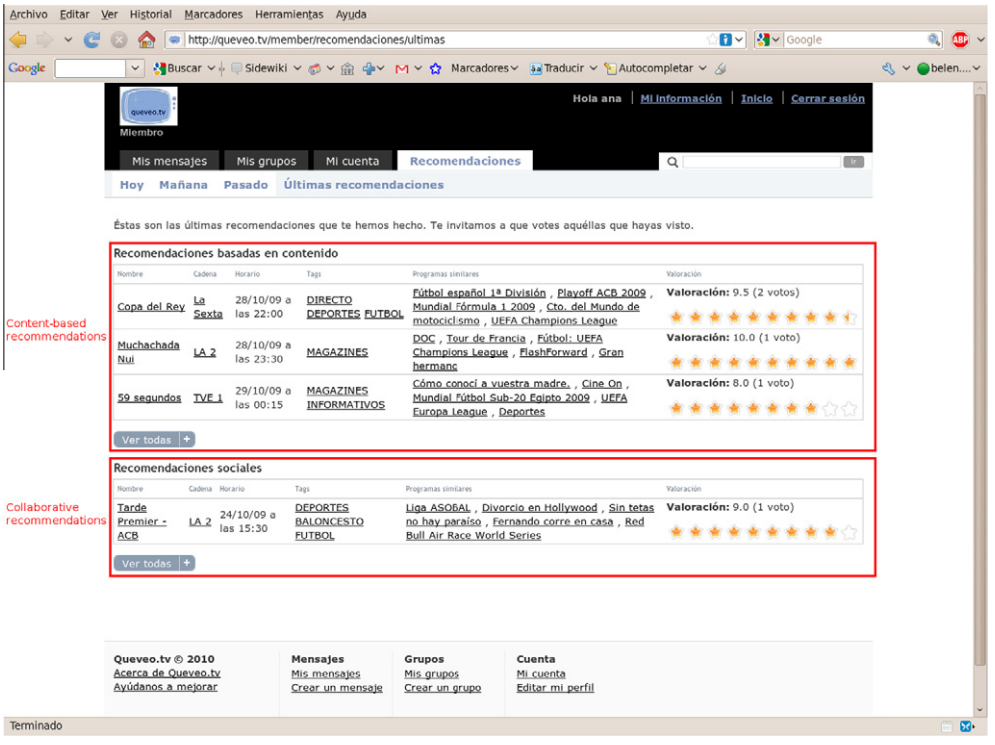

Fig. 8. First page presented to every user that has just logged into queveo.tv inviting her to rate the last recommended programs (queveo.tv snapshot).

explicitly invited to rate the last offered recommendations every time they log into the system, as we can see in Fig. 8, where each program is accompanied with grading icons allowing the user to explicitly evaluate the last proposed recommendation.

A significant design decision involves choosing the explicit rating scale. The finer grained the scale, the more information the system will have regarding each user's preference. Finer grained scales require more complex user interfaces. Each user of queveo.tv will have the opportunity to evaluate each program in a "Likert"-like scale from 1 to 10 stars (as seen in Fig. 8). Her ratings history is constantly updated although the most recent evaluations are not considered to be more important than the older ones, as in [11], because we do not believe her taste changes rapidly.

Ratings may be gathered through explicit means, implicit means, or both. Explicit ratings are those where a user is asked to provide an opinion on an item, whereas implicit ratings are those inferred from a user's actions. By necessity, our system requires explicit feedback because it is not integrated yet with a television (as happens in [50]), and cannot infer the user's interests regarding a program in question by observing the user's behavior. Explicit ratings provided by users offer the most accurate description of a user's preference for an item with the least amount of data. However, because explicit ratings require additional work from the user, it can be challenging to collect ratings, particularly when creating a new CF service, facilitating, as much as possible, this process to the user.

In this context, maintainers of CF systems sometimes use incentives to encourage users to provide more explicit ratings. For example, sites may exchange user ratings for "site points." These site points can be exchanged for rewards (e.g., t-shirts and hats) or privileges (e.g., the right to view privileged content). Researchers and practitioners [17] have proposed that users gain the following rewards from rating:

- An increased feeling of having contributed to advancing a community.
- Gratification from having one's opinions voiced and valued.
- An ability to use the CF system as an extension of their memory of what they like and dislike.

In queveo.tv, the system reminds the users that they will gain many benefits from rating other than higher quality recommendations (precisely the three elements mentioned above). While incentives may increase the number of ratings provided by users we are unaware of any studies that confirm this correlation.

The user ratings' history is used not only to predict the level of interest of a program for the active user but also to find users with similar tastes, called neighbors. This process was initially done by comparing user profiles, by measuring their similarity and retaining those profiles whose similarity level is higher than a known threshold. Our automated collaborative filtering system has been improved using a rating-based mechanism, i.e., neighbors are now selected between those who rate well the same items instead of applying measures of profiles proximity. This improvement can also be measured by a considerable increase in the values of the achieved accuracy in the set of nearest neighbors.

3.3. Program recommendation

The recommendation component is the intelligent core of queveo.tv. It must take user profile information and select new programs to recommend to a user. In Section 4 we will explain how queveo.tv uses a hybrid recommendation approach that combines content-based and collaborative recommendation strategies.

3.3.1. A content-based approach

The basic philosophy in content-based recommendation is to recommend items that match the user's likes expressed in her profile and, at the same time, which do not contain the tags used by the user to describe her dislikes. Three components are needed for content-based recommendation: (i) content descriptions for all TV programs; (ii) a compatible content description of each user's profile; and (iii) a procedure for measuring the similarity between a program and a user (see Section 4.1 for details).

3.3.2. A collaborative filtering approach

Collaborative filtering is a powerful technique that solves many of the problems associated with content-based methods. For example, there is no need for content descriptions or sophisticated similarity metrics. In fact, high quality recommendations, which would ordinarily demand a rich content representation, are possible. CF also avoids the over-specialization problem as relevant items, which are dissimilar to the items in the user profile, may appear among the recommendations, increasing their degree of novelty.

Collaborative filtering does suffer from some shortcomings. There is a startup cost associated with gathering enough profile information to make accurate user similarity measurements. There is also a latency problem due to the fact that new items will not be recommended until these items have found their way into enough user profiles. This is particularly problematic in the TV domain because new and one-off programs occur regularly and do need to be considered for recommendation. More details about the problem of recommending the so-called "one-and-only item" can be found in [13].

The key to the success of queveo.tv is the use of the combined recommendation approach. For a given user, a selection of programs is suggested, some of them are content-based recommendations, while others are collaborative recommendations. In particular, recommendation diversity is ensured through the use of collaborative filtering, and the latency problem can be solved by using content-based methods to recommend new or one-off programs. Contrary to our first impressions, collab-

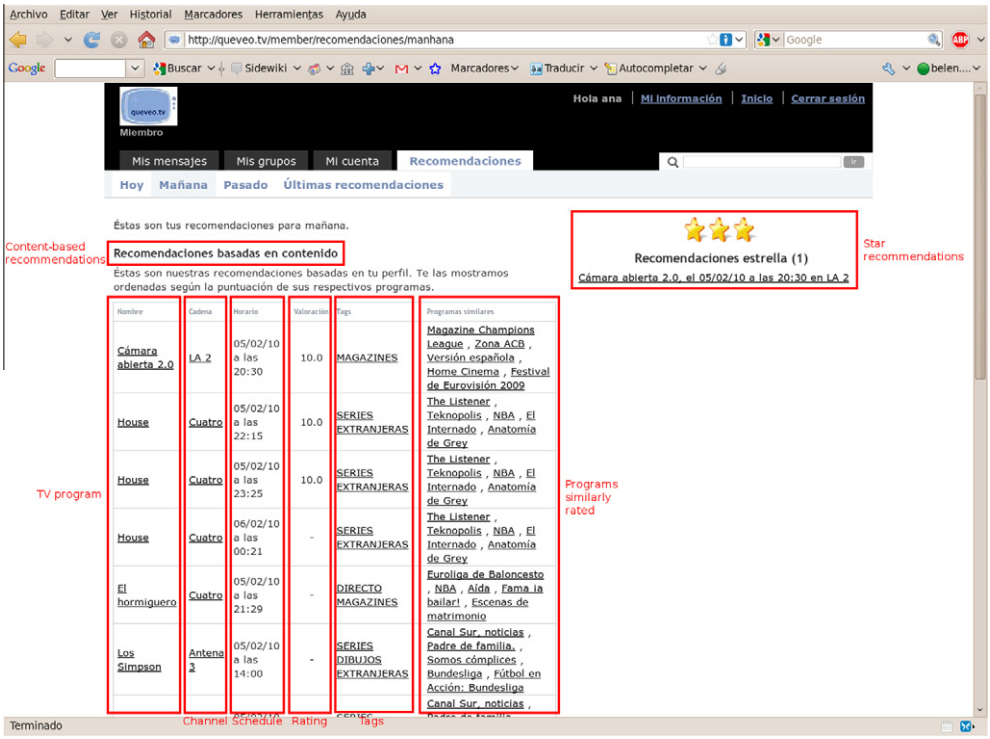

Fig. 9. Recommended shows to Ana for tomorrow. The left columns show the content-based recommendations whereas the right column lists the *star recommendations* (queveo.tv snapshot).

orative recommendations can also include one-off or new programs and this is due to the fact that many series and other kind of programs are first aired in pay-per-view or subscription-based channels, and later, broadcast in public free channels, so users of the former channels can recommend programs to users of the latter ones.

When the same program appears in both listings (content-based and collaborative recommendations), it is highlighted as a *star recommendation* because it has much more possibilities of becoming a successful recommendation.

A screenshot of queveo.tv when recommending some programs to user Ana is shown in Fig. 9. We should remark that the user will have access not only to personalized lists of programs, but also to the recommendation reasons, which explain why that particular program has been recommended. The user will be told whether the recommended programs are either content-based, or collaborative recommendations, or both, i.e., *star recommendations*. Moreover, other similar programs, in the sense that they have received the same pattern of ratings, are shown next to the recommended TV show. This is of great value to enhance overall user trust in the system [19,53,31].

4. Methodology

In the following paragraphs we will explain the methodologies followed to implement both types of filtering: content-based and collaborative.

4.1. Content-based filtering

In this section we discuss the vector space model used to generate content-based recommendations. The vector space model has been used to retrieve documents [25], and nowadays, it is often used in TV program recommendation systems [53].

The basic concept of the vector space model is the selection of an item vector which is the most similar to a retrieval key when there is one retrieval key and several item vectors need to be selected. In a TV program recommendation system, the item vectors are the number of programs and the retrieval key is the user preference. The vector space model selects a desired item by calculating the similarity between each of the item vectors and the retrieval key. Although there are several methods to calculate the similarity, such as the inner product, and the least-square method, we employ a cosine measure to calculate the similarity in our system. Formula (1) shows how the cosine correlation between the program vectors and the user model vectors is calculated.

$$sim(um, prg) = \frac{\sum_i (um(i) \times prg(i))}{\sqrt{\sum_i um(i)^2} \times \sqrt{\sum_i prg(i)^2}}. \quad (1)$$

In this formula, *um* is the user model vector and *prg* is one of the program vectors. The recommendation engine repeats this calculation for all the program vectors and sorts the programs based on the results of the calculation. The basic recommendation program listing for the user is thus obtained. This type of existing method has already been applied practically in set-top-boxes and video recorders [50]. These practical systems are able to select the programs which have the same title or cast recorded by a user in the past.

A drawback of existing TV recommendation systems is that they cannot use other user preferences to create recommendations such as those created by collaborative filtering. One of the reasons for this is that some TV recommendation systems are built into the set-top-box and cannot exchange user preferences, a thing which does not happen in our system. In the following subsection, the used collaborative filtering algorithm is explained.

4.2. Collaborative-based filtering

User-based CF systems usually take two steps:

1. Look for users who share the same rating patterns with the active user (the user whom the prediction is for).
2. Use the ratings from those like-minded users found in step 1 to calculate a prediction for the active user.

Alternatively, item-based CF proceeds in an item-centric manner:

- Build an item–item matrix determining relationships between pairs of items.
- Using the matrix, and the data on the current user, to infer her taste.

In our approach we use both methods for the achievement of different goals. On the one hand, we use the user-based approach to generate the set of neighbors for the active user (using the cosine similarity measure to find the top-*M* most similar users). This set will appear in the public user profile (as we could see in Fig. 6) enabling a user to find exactly those people who most closely share their tastes, and navigate a potentially immense social network. As stated in [43], CF web services that offer this social navigation often evolve to be much more than recommendation sites. Particularly interesting is that the CF aspects of the system would bring together communities of common interests that would then engage in direct social interaction through discussion groups, chat rooms, or email. In theory, this direct social connection is the ultimate rich interface for recommendation.

On the other hand, our system makes use of the item-based CF algorithm to generate the predictions about the level of interest of the active user for an item. In this section we will briefly discuss the item-based filtering algorithm. While user-based algorithms generate predictions based on similarities between users, item-based algorithms generate predictions based on similarities between items. The prediction for an item should be based on a user's ratings for similar items.

The similarity function used by the nearest neighbor algorithms depends on the type of data. For structured data, an Euclidean distance metric is often used. When using the vector space model (as seen in Section 4.1), the cosine similarity measure is often used. In the Euclidean distance function, the same feature having a small value in two examples is treated the same as that feature having a large value in both examples. In contrast, the cosine similarity function will not have a large value if corresponding features of two examples have small values. As a consequence, it is appropriate for text when we want two documents to be similar only if they are about the same topic, but not if they are both not about a topic [33].

The execution steps of the algorithm are:

(i) *Data representation* of the ratings provided by m users on n items. This step is based on the construction of an $m \times n$ user-item matrix, *R*.
(ii) *Neighborhood formation*, which concludes by the construction of the active item's neighborhood. Similarities for all possible pairs of items existing in the data set are computed by the application of the preferred similarity metric (in our case, by using the adjusted cosine similarity measure). Items most similar to the active item, which refers to the item for which predictions should be generated, are selected for its neighborhood.
(iii) *Prediction generation*, where the final predictions are calculated as a weighted sum of ratings given by a user on all items included in the active item's neighborhood.

5. SVD technology

Several algorithms reduce domain complexity by mapping the item space to a smaller number of underlying "dimensions". Intuitively, these dimensions might represent the latent topics or tastes of those items. The smaller "latent" dimensions reduce runtime performance needs and lead to larger numbers of co-rated dimensions. These techniques define a mapping between a user's ratings and their underlying tastes. An item's prediction can then be generated based on a user's

underlying tastes. Mapping functions generally consist of simple vector operations, and predictions for an item can be calculated in constant time. Vector-based techniques for extracting underlying dimensions include support vector decomposition [41,55], and principal component analysis [15].

The algorithm used in queveo.tv utilizes the well-known matrix factorization technique called singular value decomposition (SVD) to implement the item-based collaborative filtering. Our proposal uses SVD in order to reduce the dimension of the active item's neighborhood, and then it executes the item-based filtering with this low-rank representation to generate its predictions.

5.1. Singular value decomposition

Singular value decomposition (SVD) [16] is a matrix decomposition technique which takes an $m \times n$ matrix R, with rank r, and decomposes it as follows:

$$SVD(R) = U \times S \times V^T. \tag{2}$$

U and V are two orthogonal matrices with dimensions $m \times m$ and $n \times n$, respectively. S, called the *singular matrix*, is an $m \times n$ diagonal matrix whose diagonal entries are non-negative real numbers.

The initial r diagonal entries of S $(s_1, s_2, \ldots, s_r)$ have the property that $s_i > 0$ and $s_1 \geqslant s_2 \geqslant \cdots \geqslant s_r$. Accordingly, the first r columns of U are eigenvectors of RR^T and represent the left singular vectors of R, spanning the column space. The first r columns of V are eigenvectors of R^TR and represent the right singular vectors of R, spanning the row space. If we focus only on these r nonzero singular values, the effective dimensions of the SVD matrices U, S and V will become $m \times r$, $r \times r$ and $r \times n$ respectively.

An important attribute of SVD, particularly useful in the case of recommender systems, is that it can provide the best low-rank approximation of the original matrix R. By retaining the first singular values of S and discarding the rest, which can be translated as keeping the *k largest* singular values, based on the fact that the entries in S are sorted, we reduce the dimensionality of the data representation and hope to capture the important "latent" relations existing but not evident in the original representation of R. The resulting diagonal matrix is termed S_k. Matrices U and V should be also reduced accordingly. U_k is produced by removing $r - k$ columns from matrix U. V_k is produced by removing $r - k$ rows from matrix V. Matrix R_k which is defined as:

$$R_k = U_k \times S_k \times V_k^T, \tag{3}$$

stands for the closest linear approximation of the original matrix R with reduced rank k. Once this transformation is completed, users and items can be represented as points in the k-dimensional space.

5.2. Our CF algorithm enhanced with SVD

We will now describe the steps to combine SVD with item-based filtering in order to make the base algorithm more scalable following the proposal in [55,57]:

1. **Initial matrix** R. We define the original user-item matrix, R, of size $m \times n$, which includes the ratings of m users on n items. r_{ij} refers to the rating of user u_i on item i_j.
2. **Filling and normalization of** R. We preprocess user-item matrix R in order to eliminate all missing data values. The preprocessing is described in detail next:
   (a) Compute the average of all rows, r_i, where $i = 1, 2, \ldots, m$, and the average of all columns, r_j, where $j = 1, 2, \ldots, n$, from the user-item matrix, R.
   (b) Replace all matrix entries that have no values (those programs not rated yet) with the corresponding column average, r_j, which leads to a new filled-in matrix, $R_{filled-in}$.
   (c) Subtract the corresponding row average, r_i, from all the slots of the new filled-in matrix, $R_{filled-in}$, and obtain the normalized matrix R_{norm}. This normalization step allows the values in R_{norm} to be independent of the way the users rate.
3. **SVD computation**. We compute the SVD of R_{norm} and obtain matrices U, S, and V, of size $m \times m$, $m \times n$, and $n \times n$, respectively. Their relationship is expressed by: $R_{norm} = U \times S \times V^T$.
4. **Dimensionality reduction**. We perform the dimensionality reduction step by keeping only k diagonal entries from matrix S to obtain a $k \times k$ matrix, S_k. Similarly, matrices U_k and V_k of size $m \times k$ and $k \times n$ are generated. The "reduced" user-item matrix, R_{red}, is obtained by $R_{red} = U_k \times S_k \times V_k^T$, while rr_{ij} denotes the rating by user u_i on item i_j as included in this reduced matrix.
5. **Representation of users and programs in a** k**-dimensional space**. We compute $\sqrt{S_k}$ and then calculate two matrix products:
   - $U_k \times \sqrt{S_k}^T$, which represents m users in the k dimensional feature space, and
   - $\sqrt{S_k} \times V_k^T$, which represents n items in the k dimensional feature space.

We are particularly interested in the latter matrix, of size $k \times n$, whose entries represent the "meta" ratings provided by the k pseudo-users on the n items. A "meta" rating assigned by pseudo-user u_i on item i_j is denoted by mr_{ij}.

6. **Neighborhood formation**. We proceed with neighborhood formation which can be broken into two substeps:
   (a) Calculate the similarity between items i_j and i_f by computing their adjusted cosine similarity as follows:

$$sim_{jf} = adjcorr_{jf} = \frac{\sum_{i=1}^{k} mr_{ij} \cdot mr_{if}}{\sqrt{\sum_{i=1}^{k} mr_{ij}^2 \cdot \sum_{i=1}^{k} mr_{if}^2}} \tag{4}$$

where k is the number of pseudo-users, selected when performing the dimensionality reduction step. We have to note a change between the adjusted cosine similarity equation utilized in plain item-based filtering and here. In plain item-based filtering the difference in rating scale between distinct users was offset by subtracting the corresponding user average from each co-rated pair of items. In SVD-enhanced item-based filtering, that difference in rating scale was offset during the normalization of the original user-item matrix which yielded matrix R_{norm}.

   (b) Based on the results from the adjusted cosine similarity calculations for pairs of items, including the active item and a random item, isolate the set of items which appear to be the most similar to the active item.

7. **Generate the predictions**. We conclude with prediction generation, achieved by the following weighted sum:

$$pr_{aj} = \frac{\sum_{k=1}^{l} sim_{jk} \cdot (rr_{ak} + \overline{r_a})}{\sum_{k=1}^{l} |sim_{jk}|} \tag{5}$$

which calculates the prediction for user u_a on item i_j. It is similar to the equation utilized by plain item-based filtering in that it bases its predictions on the ratings given by the active user, u_a, on the l items selected as the most similar to active item i_j. Yet, it is different in that the user ratings are taken from the reduced user-item matrix, R_{red}. Also, we have to add the original user average, $\overline{r_a}$, back since it was subtracted during the normalization step of the preprocessing.

Once obtained the predictions, the algorithm returns to the active user the set of programs satisfying the two following conditions:

(1) the prediction exceeds a predefined threshold (we are currently using a threshold of 7), and
(2) the program has not been seen (rated) yet by the active user (i.e., with an empty value in the initial user-program matrix R).

5.3. Benefits of applying SVD

As explained in [55], a recommender system running item-based filtering with a lower dimensional representation, as the previously described, will benefit in the following ways:

- The complexity of item-based filtering, utilizing the original data representation, is $O(mn^2)$. By reducing the dimension to k, where $k \ll m$, the complexity becomes $O(kn^2)$. We can assume that this reduction in complexity will improve the scalability of the system, while both the processing time and storage requirement should also move down.
- Based on the properties of SVD, any latent relations between users and items should be located when employing the low rank data representation.
- Before the main part of the algorithm is executed, during its preprocessing phase, all the empty entries of the user-item matrix are filled. As a result, once the execution is completed, the n items, taken from the original data array, have now been rated by all, k, pseudo-users. This means that the sparsity problem is solved and the achieved coverage for the recommender system is always equal to 100%.

Still, we are interested to find out if the benefits for applying item-based filtering on a low dimensional neighborhood are also extended to the accuracy of the generated predictions. To achieve that we have set up a number of experiments which will be discussed in detail in Section 6.

6. Evaluation

Evaluation is a core aspect of recommender systems design and deployment. The aspects generally evaluated are the accuracy and the coverage of a system's recommendation algorithms [20]. Nevertheless, Herlocker et al. [21] outlined the importance that other system aspects also have, and proposed the consideration of measures related to the suitability of recommendations to users (e.g., confidence in the recommendation, learning rate, novelty/serendipity), the satisfaction of the users, and the technical performance of the system. Those techniques have been divided in [21] into three categories: (i) *predictive accuracy metrics*, which measure how close the recommender's predictions are to the true user ratings; (ii) *classification accuracy metrics*, which measure how often a recommender system can decide correctly whether an item is beneficial for the user and therefore should be suggested to her; and (iii) *rank accuracy metrics*, which measure the proximity of a predicted ordering of items, as generated by a recommender system, to the actual user ordering of the same items.

As stated in [21], the choice among these metrics should be based on the selected user tasks and the nature of the data sets. We wanted our proposed algorithms to derive a predicted score for already rated items rather than generate a top-*N* recommendation list. Based on that specific task, we selected *mean absolute error* (MAE) as the appropriate evaluation metric for our experiments. MAE is a statistical accuracy metric that measures the deviation of predictions generated by the recommender system from the true rating values as they were specified by the user. Therefore, MAE is measured only for those items for which a user has expressed her opinion.

For the execution of our subsequent experiments we utilized the data publicly available from the GroupLens movie recommender system. The MovieLens data set consists of 100.000 ratings which were assigned by 943 users on 1682 movies. Users should have stated their opinions for at least 20 movies in order to be included. Ratings follow the 1 (bad) to 5 (excellent) numerical scale. Starting from the initial data set, five distinct splits of training and test data were generated.

The aim of the experiment is to locate the optimal parameter settings for the item-based enhanced by SVD filtering algorithm, and to see what MAE value we obtain for this optimal configuration.

6.1. Locating the optimal value for reduced dimension, k

As mentioned earlier, k refers to the number of singular values retained from the singular matrix S. As a result, k also corresponds to the rank of the reduced user-item matrix R_{red}, and also to the number of pseudo-users which are considered, instead of the actual m users, in order to represent the n items.

The number of dimensions selected for the reduced space representation, as expressed by the value of k, is significant for the efficiency of the recommender system which incorporates this representation for its data. On the one hand, the number of dimensions should be rather small in order to actually lead to an improvement in the filtering algorithm's scalability and, at the same time, to exclude any over-fitting errors. On the other hand, it should be big enough in order to capture any latent relations among the users or the items included in the original data matrix, R.

By this experiment we wanted to determine the ideal value of this dimension. We kept the size of the active item's neighborhood fixed to 60 (as in [55]), and ran our algorithm repeatedly for different values of k, $k = \{2,4,6,8,10,11,12,13,14,15,16,17,18,19,20,22,24,26,28,30,35,40,45,50\}$, which is a bigger set of values than in [55] where authors used only eight values of k. Fig. 10 collects the mean absolute errors observed by those runs, while the averaged over the five data splits from the data set is shown in Fig. 11.

Our results indicate that specifically, at first, the predictions' accuracy improves as we increase the rank of the lower dimension space and quickly reaches its peak for $k = 11$ (for three sets) or $k = 16$ (for two data sets). For any further increase in the rank, the results keep getting worse. It appears that the size of the data set and its sparsity allow for existing latent relations among items to be discovered for rather low rank values, while the over-fitting of the data is evident when the value of k increases. Our value of optimal k is a little bit higher than in [55] where authors obtained as optimal $k = 6$.

Regarding the accuracy, our experiment shows that, once configured the parameters of the collaborative filtering ($k = 12$), we obtain a MAE value of 0.78. This metric is traditionally used to represent the ability of the approach on narrowing the gap between user's preferences and the automated recommendations (as it is also pointed out in [23]). That is, the lower the MAE, the lower the gap, and the more accurate the predictions. In this case, a MAE value of 0.78 means that if the predicted value for an active user and a program is 8.5 (in our scale from 1 to 10), the true rating given by this user (to that program) would be in the interval [7.72, 9.28]. By comparing our result (0.78) with other MAE values in related work (as [55,56], where

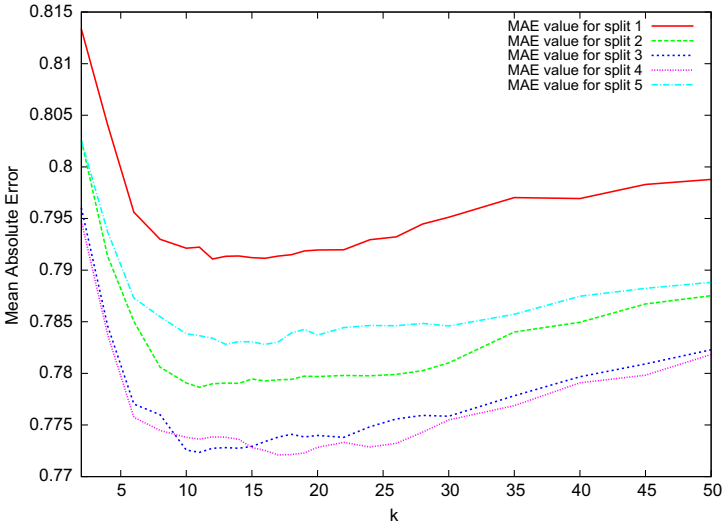

Fig. 10. MAE values for the five sets of data.

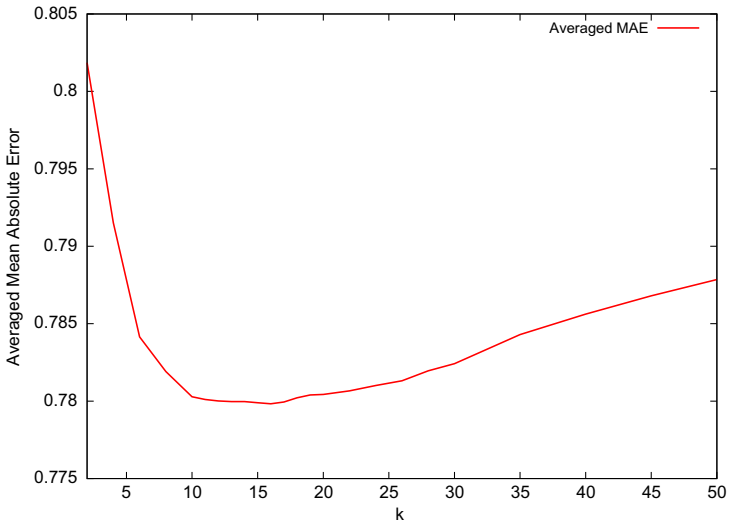

Fig. 11. Averaged MAE.

the results for MAE were 0.7965 and 0.8398, for item-based filtering with low-dimensional and high-dimensional neighborhood, respectively), we can conclude this is a reasonably good result for accuracy, and we can be quite sure that the recommendations suggested by queveo.tv fit quite well with users' likes.

Regarding the value of the active item's neighborhood (l), it was set to $l = 60$ to be able to compare our results with those in [55]. However, in the daily execution of our recommender engine, we have experimented with different values for l, ranging from 10 to 25. As a result, we have surprisingly obtained more accurate results for a neighborhood including only 10 items. This can confirm the findings of the experiments in [55,56], having that item-based filtering reaches its peak performance for quite smaller neighborhoods than those required for similar experiments in user-based filtering.

6.2. Future work in evaluation

Although many recent works continue using the MAE measure (for example, [46]), it also is well-known that it has received criticism as well [48]. The first reason why we have used the MAE value to assess our results was to make it possible the comparison with other previous works, because most of them had used this metric (e.g., [18,41,42,4,55]). The second reason is that many works ([42,29,55] among others) recommend to use this measure when evaluating an item-based CF algorithm (where the interest for each item is predicted), while other metrics are more suitable for assessing those algorithms that generate a top-N recommendation list. In this latter case, rank accuracy metrics attempt to compute the utility of a recommendation list to a user. Common rank accuracy metrics include precision [29,41] and half-life utility [10].

However, it is also true that we need to complete our evaluation in several ways. Besides prediction accuracy, our approach is also concerned with the recommendation coverage (the number of programs whose prediction is computed) as well as the time required to make run-time predictions in realistic systems.

We believe that in a TV system, coverage is very important because most TV programs are only available at the moment of broadcasting. This means that we are interested in both prediction quality and coverage at the same time. For this reason, we are planning to combine both measures into the new measure used in [54]: the *Global Mean Absolute Error* (GMAE). This measure is the same as MAE, but when a prediction cannot be made the neutral value is assumed (which is how users probably see a non prediction).

In addition, in traditional information retrieval domain, system's quality is evaluated by *recall* and *precision*. For TV program recommendation systems, their quality can be also appraised by these two parameters, but some modification to them is needed, as explained in [60]. Given a time interval, let *watched* denote the program set which the user has watched in the interval and *recommended* denote the program set which system has recommended in the same time interval, thus the *recall* and *precision* can be defined as follows:

$$recall = \frac{|watched \cap recommended|}{|watched|}, \tag{6}$$

$$precision = \frac{|watched \cap recommended|}{|recommended|}. \tag{7}$$

These two measures are, however, often conflicting in nature, for instance, increasing the number of recommendations tends to increase *recall* but decrease *precision*. The fact that both are critical for quality judgment leads to the combination of the two, giving equal weight to each, the system's quality can be evaluated by $F1$ [7]:

$$F1 = \frac{2 \times recall \times precision}{recall + precision}. \quad (8)$$

We are also planning to improve our evaluation experiments using the $F1$ measure.

7. Implementation details and deployment

It is our contention that the correct way to provide a truly useful and scalable solution to the TV listings problem is to offer a personalized service. Our objective was to produce a web 2.0 application that could offer individual users online TV shows recommendations that were carefully tailored to match the viewing preferences of each target user. Moreover, the personalization would be carried out automatically using a combination of user profiling, content-based, and collaborative filtering techniques.

7.1. Hardware and software

The development of queveo.tv is fully based on open-source technologies. We developed this web application by using the scripting language *Ruby* and the framework *Rails*, while the databases of users' profiles, programs, broadcastings, groups, ratings, comments, and TV channels are implemented in *MySQL*. queveo.tv server currently runs on Ubuntu Jaunty Linux on Intel (R) Core (TM)2 Quad CPU Q9300 at 2.50 GHz with 4 GB of RAM and has been stress-tested beyond 20,000 hits in average per month without any performance degradation.

For gathering the complete listings of all TV channels, the application makes use of XMLTV [59], which is a set of programs to process TV listings, storing them in an XMLTV format (based on XML). In a first version of queveo.tv, we made use of available backends to download TV listings, filter programs and *Perl* libraries to process those listings, but finally it was more practical to design and implement our own parser of TV listings downloaded from different specialized web sources.

In order to compute the singular value decomposition, we use *linalg* [28], which is a fast LAPACK-based library for real and complex matrices. Precisely, to alleviate the response time invested in computing new recommendations for a given user, the predictions matrix is updated (to take into account the new ratings) offline every 10 min (this parameter is configurable depending on the application needs). To this end, we have separately implemented a SVD server by using a devoted process, which guarantees the consistency of the data matrix by ensuring that each access to modify any value is done in mutual exclusion. Finally, to add social network features and to improve the web presentation, queveo.tv makes use of *tog* [51], a non-intrusive, extensible open source platform specialized in creating communities and giving social capabilities to websites.

7.2. Deployment

queveo.tv has been deployed in April 2009 in a production environment making use of *nginx*: a fast, lightweight, and efficient web server. The number of users has grown to nearly 1000 with about 40,000 recommendation lists generated per month. Furthermore, queveo.tv has not been publicized, thus its current popularity is based largely on word of mouth and unsolicited press coverage. In fact, since its launch, the site has received many favorable press reviews from international science news websites as Science *Daily* [37], PhysOrg.com [36], and ACM Tech News [34], Spanish magazines and newspapers including, e.g., PC World [35], and several important websites (in Spanish) specialized in TV listings, e.g., vertele.com [38].

For unregistered users or users entering queveo.tv for the first time, the system presents itself as a retrieval system. Its functionality at this stage restricts itself to the functionality of a printed TV program guide, with a graphical user interface. Users specify a query, sort through the resulting list and select programs to watch. They can also see the TV schedule for "today", "tomorrow", and "the day after tomorrow" for more than 130 TV channels. We consider that offering in advance the information for three days is enough because most of users only plan a TV schedule for the following day or they did not plan at all. In fact, many users only use a guide to determine what is currently on TV [8].

Additionally, for registered users, queveo.tv shows the recommended TV schedule for "today", "tomorrow", and "the day after tomorrow" according to the preferences specified in the user profile and the ratings given by her and her similar users. queveo.tv also allows users who serve as self-proclaimed editors to give tips that contain recommendations volunteered by these so-called *opinion leaders*. queveo.tv offers the users the possibility of creating groups focused on preferred programs, particular tastes as terror films, realities, etc., because, as evidenced by the popularity of message boards relating to TV shows and current events, people often want to comment on the content of mass-media broadcasts. It also favors the communication among users: each user can send/receive private messages through our platform. This way, queveo.tv provides an ad hoc social community including viewers watching the same show on TV.

The current version of queveo.tv is in Spanish, but it can be easily extended to many other languages and countries using backends to download TV listings for several countries or developing our own parsers. We are currently validating this version with 857 users, 9367 programs, 133 TV channels, 340, 181 broadcastings, 43 groups, 3610 ratings, and 210 comments. We plan to extend the functionalities adding the possibility of creating series' blogs where fans can comment their favorite

episodes, allowing users to add more content to the application (e.g., images and tags describing programs), to correct mistakes in the TV listings, etc. This will result in a more accurate programming information indeed.

So far, we have received numerous comments and feedback from the users through the web application. Most of them are extremely positive. Critically, only 10% of messages report some problem with the rating process, since people are used to vote a program, not an only broadcasting. We have explained them that we have used this method to be able to distinguish, e.g., among different episodes of a TV series, and we decided to follow the way in which series are rated in many well-known websites like IMDb (*Internet Movie Database* [49]). 90% of these users found the site to be easy to use as a source of TV recommendations.

8. Conclusions and future work

In the following subsections, we present the main conclusions of our work as well as the ongoing and future research.

8.1. Conclusions

Summarizing, queveo.tv offers an effective solution to the very real problem of providing people with relevant TV listings information. Our web 2.0 application personalizes TV information to meet the viewing preferences of individual users. It is the one, to the best of our knowledge, which successfully integrates content-based and collaborative filtering techniques, providing efficient solutions to the problems of both algorithms (particularly, the sparsity and scalability problems). Besides, queveo.tv provides all typical features of any social network, such as supporting communication among users as well as allowing users to add and tag contents, rate and comment the items, among other functionalities. This way, queveo.tv provides an ad hoc social community: the perfect meeting point for people to enjoy themselves, sharing their opinions and ratings about their favorite TV shows.

With respect to performance implications, we should note that some tasks that require the biggest amount of computation (downloading the listings and storing them in databases) are computed offline. The online component dynamically computes to provide predictions to customers using data from the component stored offline. To reduce this computation overload, we gave a detailed description of the way SVD can be utilized in order to reduce the dimension of the user-item representation, and afterwards, how this low-rank representation can be employed in order to generate item-based predictions.

A number of experiments were set up to check the validity of our CF algorithm. The results showed that low-dimension item-based filtering not only alleviates problems like scalability or sparsity of the data, but also proves to be a very accurate recommender. We have also tested our application with real users in order to get an appraisal of our personalization techniques and we have received very much positive feedback.

8.2. Future work

An obvious future work is determined by the need of an effective and efficient management of a big amount of data. It is likely that, in a near future, we must face the problem of having an overall volume of data which is highly increased so we will need to scale more our computations. To cope with this, we are studying the possibility of parallelizing our algorithms in a Hadoop [3] framework (using the cloud computing paradigm).

As part of our future work regarding the evaluation, we have already stated in Section 6.2 some of our immediate goals. Aside from this, we would like to measure how users perceive the system and the recommendations' quality which can be accomplished either by surveying the users [47,32] or measuring retention and use statistics. Regarding to the latter, and according to the data managed by Alexa Web Crawler [52], each user spends 5, 1 min per day on average visiting queveo.tv. More use statistics including traffic data, related links and more should be considered to analyze growth and understand the effects of specific events on our web site traffic. Additionally, in order to complete our evaluation, it is in our future plans to design a complete survey which is both system-centric and user-centric as possible. We would also like to repeat the study, carried out in Section 6 using the MovieLens data collection, with our own collected data to check whether the values of MAE and the optimal k continue being similar.

Other future research is related to social tagging. Currently, our application uses tags to describe each program' contents and users' likes and hates. With this information, queveo.tv provides functionalities like the navigation among all the profiles of users who share a particular tag to describe their likes/dislikes, or among all TV content tagged with given tags. In this context, we would like to propose a social tagging system that allows users to add additional tagging information to enrich each TV program description. This would also allow us to incorporate some improvements to our algorithms, enriching the results with the ones provided by a tag-based recommender which suggests to a user those elements which have been tagged by other users who share tags with her.

Moreover, it is also remarkable the vast penetration of high-end mobile devices together with the introduction of flat rate data plans from many popular mobile operators which have resulted in larger usage of mobile services. This has created new opportunities for making the applications and services available on these devices more intelligent and supportive to the user. This is why we also plan to extend the service to be offered to both handheld devices and set-top-boxes. Once the profile is

filled in using the web interface, it is also valuable that users can query the recommendations sat in their living room by using their smartphone or set-top-box with Internet access. And even more, the near future would be able to see the queveo.tv service being delivered directly to our television sets via Internet enabled PVRs (Personal Video Recorders), in such a way that makes possible the automatic recording of our recommended programs.

Finally, we should remark that the underlying personalization technology of queveo.tv is not limited at all to personalizing TV listings. We are currently undertaking the design and development of a system that generates recommendations about local events in the city of Vigo to be delivered to a mobile device. This system provides a list of activities (represented in a map) which are recommended by taking into account both the user's preferences and the ratings of other similar users, i.e., utilizing the same recommendation technology used in queveo.tv. Aside from that, this service takes into account contextual information (location, time, weather, etc.) to filter the recommendations and also provides details on how and when to perform those offered activities, for example, giving GPS guidance to the points of interest.

Acknowledgments

Thanks should be given to Alberto Lago and Óscar Álvarez, two undergraduate students who have contributed to the implementation of the application. The authors also thank the anonymous reviewers and the Editor-in-Chief for valuable remarks and suggestions.

References

[1] G. Adomavicius, R. Sankaranarayanan, S. Sen, A. Tuzhilin, Incorporating contextual information in recommender systems using a multidimensional approach, ACM Transactions on Information Systems 23 (1) (2005) 103–145.
[2] G. Adomavicius, A. Tuzhilin, Toward 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) (2005) 734–749.
[3] Apache Hadoop Project, 2010. URL: <http://hadoop.apache.org>.
[4] L. Ardissono, C. Gena, P. Torasso, F. Bellifemine, A. Difino, B. Negro, User modeling and recommendation techniques for personalized electronic program guides, in: Personalized Digital Television – Targeting Programs to Individual Viewers, Human–Computer Interaction Series, vol. 6, Kluwer Academic Publishers, 2004, pp. 3–26 (Chapter 1).
[5] M. Balabanovic, Y. Shoham, Fab: content-based, collaborative recommendation, Communications of the ACM 40 (3) (1997) 66–72.
[6] A.B. Barragáns Martínez, J. Pazos Arias, A. Fernández Vilas, J. García Duque, M. López Nores, What's on TV tonight? An efficient and effective personalized recommender system of TV programs, IEEE Transactions on Consumer Electronics 55 (1) (2009) 286–294.
[7] C. Basu, H. Hirsh, W. Cohen, Recommendation as classification: using social and content-based information in recommendation, in: Proceedings of the Recommender System Workshop at the 15th National Conference on Artificial Intelligence (AAAI'98), 1998, pp. 714–720.
[8] P. Baudisch, Dynamic Information Filtering, Ph.D. Thesis, GMD Research Series, No. 16, GMD Forschungszentrum Informationstechnik GmbH, Sankt Augustin, 2001.
[9] P. Baudisch, L. Brueckner, TV scout: lowering the entry barrier to personalized TV program recommendation, in: P.D. Bra, P. Brusilovsky, R. Conejo (Eds.), Proceedings of Adaptive Hypermedia and Adaptive Web-Based Systems (AH'02), Lecture Notes in Computer Science, vol. 2347, Springer, 2002, pp. 58–68.
[10] J.S.Breese, D. Heckerman, C. Kadie, Empirical analysis of predictive algorithms for collaborative filtering, in: Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, 1999, pp. 43–52.
[11] D. Bueno, R. Conejo, D. Martín, J. León, J.G. Recuenco, What can i watch on TV tonight, in: Wolfgang Nejdl et al. (Eds.), Proceedings of Adaptive Hypermedia and Adaptive Web-Based Systems (AH'08), Lecture Notes in Computer Science, vol. 5149, Springer, 2008, pp. 271–274.
[12] R. Burke, Hybrid recommender systems: survey and experiments, User Modeling and User-Adapted Interaction 12 (4) (2002) 331–370.
[13] C. Cornelis, J. Lu, X. Guo, G. Zhang, One-and-only item recommendation with fuzzy logic techniques, Information Sciences 177 (22) (2007) 4906–4921.
[14] M. Ehrmantraut, T. Härder, H. Wittig, R. Steinmetz, The personal electronic program guide—towards the pre-selection of individual TV programs, in: Proceedings of the 5th International Conference on Information and Knowledge Management (CIKM'96), ACM Press, 1996, pp. 243–250.
[15] K. Goldberg, T. Roeder, D. Gupta, C. Perkins, Eigentaste: a constant time collaborative filtering algorithm, Information Retrieval 4 (2) (2001) 133–151.
[16] G. Golub, C.V. Loan, Matrix Computations, third ed., Johns Hopkins Studies in Mathematical Sciences, Baltimore, 1996.
[17] M. Harper, S.X. Li, Y. Chen, J.A. Konstan, Social comparisons to motivate contributions to an online community, in: Proceedings of the 2nd International Conference on Persuasive Technology (PERSUASIVE'07), 2007, pp. 148–159.
[18] J.L. Herlocker, J.A. Konstan, A. Borchers, J.T. Riedl, An algorithmic framework for performing collaborative filtering, in: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM Press, 1999, pp. 230–237.
[19] J.L. Herlocker, J.A. Konstan, J.T. Riedl, Explaining collaborative filtering recommendations, in: Proceedings of the ACM Conference on Computer Supported Cooperative Work, 2000, pp. 241–250.
[20] J.L. Herlocker, J.A. Konstan, J.T. Riedl, An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms, Information Retrieval 5 (4) (2002) 287–310.
[21] J.L. Herlocker, J.A. Konstan, L.G. Terveen, J.T. Riedl, Evaluating collaborative filtering recommender systems, ACM Transactions on Information Systems 22 (1) (2004) 5–53.
[22] S.H. Hsu, M.-H. Wen, H.-C. Lin, C.-C. Lee, C.-H. Lee, AIMED – a personalized TV recommendation system, in: Pablo César, Konstantinos Chorianopoulos, et al. (Eds.), Interactive TV: A Shared Experience, Lecture Notes in Computer Science, vol. 4471, Springer, 2007, pp. 166–174.
[23] B. Jeong, J. Lee, H. Cho, Improving memory-based collaborative filtering via similarity updating and prediction modulation, Information Sciences 180 (5) (2010) 602–612.
[24] K. Kurapati, S. Gutta, D. Schaffer, J. Martino, J. Zimmerman, A multi-agent TV recommender, in: Proceedings of the UM 2001 Workshop Personalization in Future TV, 2001.
[25] D.L. Lee, H. Chuang, K.E. Seamons, Document ranking and the vector-space model, IEEE Software 14 (2) (1997) 67–75.
[26] G. Lekakos, G.M. Giaglis, Improving the prediction accuracy of recommendation algorithms: approaches anchored on human factors, Interacting with Computers 18 (3) (2006) 410–431.
[27] D. Lemire, A. Maclachlan, Slope one predictors for online rating-based collaborative filtering, in: Proceedings of SIAM Data Mining (SDM'05), 2005.
[28] Linalg Project Home Page, 2010. URL: <http://rubyforge.org/projects/linalg>.
[29] M.R. McLaughlin, J.L. Herlocker, A collaborative filtering algorithm and evaluation metric that accurately model the user experience, in: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM Press, 2004, pp. 329–336.
[30] B. Mobasher, Data mining for web personalization, in: P. Brusilovsky, A. Kobsa, W. Nejdl (Eds.), The Adaptive Web, Lecture Notes in Computer Science, vol. 4321, Springer-Verlag, 2007, pp. 90–135.

[31] J. O'Donovan, B. Smyth, B. Gretarsson, S. Bostandjiev, T. Höllerer, PeerChooser: visual interactive recommendation, in: Proceedings of the 26th SIGCHI Conference on Human Factors in Computing Systems (CHI'08), ACM Press, New York, NY, USA, 2008, pp. 1085–1088.
[32] C.-S. Ong, M.-Y. Day, W.-L. Hsu, The measurement of user satisfaction with question answering systems, Information and Management 46 (7) (2009) 397–403.
[33] M.J. Pazzani, D. Billsus, Content-based recommendation systems, in: P. Brusilovsky, A. Kobsa, W. Nejdl (Eds.), The Adaptive Web, Lecture Notes in Computer Science, vol. 4321, Springer-Verlag, 2007, pp. 325–341.
[34] queveo.tv at ACM Tech News, 2010. URL: <http://cacm.acm.org/news/29279-web-20-application-developed-to-recommend-television-programs/fulltext>.
[35] queveo.tv at PC World, 2010. URL: <http://www.idg.es/pcworld/La-Universidad-de-Vigo-crea-el-Recomendador-de-TV/doc80951-TV.htm>.
[36] queveo.tv at PhysOrg.com, 2010. URL: <http://www.physorg.com/news162468111.html>.
[37] queveo.tv at Science Daily, 2010. URL: <http://www.sciencedaily.com/releases/2009/05/0905251>.
[38] queveo.tv at vertele.com, 2010. URL: <http://www.vertele.com/noticias/detail.php?id=22953>.
[39] queveo.tv Project Home Page, 2010. URL: <http://www.queveo.tv>.
[40] P. Resnick, N. Iacovou, M. Suchack, P. Bergstrom, J.T. Riedl, GroupLens: an open architecture for collaborative filtering of netnews, in: Proceedings of the ACM Conference on Computer Supported Cooperative Work, 1994, pp. 175–186.
[41] B.M. Sarwar, G. Karypis, J. Konstan, J.T. Riedl, Application of dimensionality reduction in recommender system – a case study, in: Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Workshop on Web Mining for E-Commerce – Challenges and Opportunities (WEBKDD'00), Boston, Massachusetts, USA, 2000.
[42] B.M. Sarwar, G. Karypis, J. Konstan, J.T. Riedl, Item-based collaborative filtering recommendation algorithms, in: Proceedings of the 10th International Conference on World Wide Web (WWW'01), ACM, Boston, Massachusetts, USA, 2001, pp. 285–295.
[43] J.B. Schafer, D. Frankowski, J. Herlocker, S. Sen, Collaborative filtering recommender systems, in: P. Brusilovsky, A. Kobsa, W. Nejdl (Eds.), The Adaptive Web, Lecture Notes in Computer Science, vol. 4321, Springer-Verlag, 2007, pp. 291–324.
[44] B. Smyth, P. Cotter, A personalized television listing service, Communications of the ACM 43 (8) (2000) 107–111.
[45] B. Smyth, P. Cotter, A personalized TV listings service for the digital TV age, Knowledge-Based Systems 13 (2–3) (2000) 53–59.
[46] J.-H. Su, B.-W. Wan, C.-Y. Hsiao, V.S. Tseng, Personalized rough-set-based recommendation by integrating multiple contents and collaborative information, Information Sciences 180 (1) (2010) 113–131.
[47] K. Swearingen, R. Sinha, Beyond algorithms: an HCI perspective on recommender systems, in: Proceedings of the ACM SIGIR 2001 Workshop on Recommender Systems, ACM Press, 2001.
[48] P. Symeonidis, A. Nanopoulos, A.N. Papadopoulos, Y. Manolopoulos, Collaborative recommender systems: combining effectiveness and efficiency, Expert Systems with Applications 34 (4) (2008) 2995–3013.
[49] The Internet Movie Database, 2010. URL: <http://www.imdb.com>.
[50] TiVo, Inc. 2002. TiVoTM Home Page, 2010. URL: <http://www.tivo.com>.
[51] Tog Project Home Page, 2010. URL: <http://www.toghq.com>.
[52] Traffic details about queveo.tv from Alexa, 2010. URL: <http://www.alexa.com/siteinfo/queveo.tv>.
[53] T. Tsunoda, M. Hoshino, Automatic metadata expansion and indirect collaborative filtering for TV program recommendation system, Multimedia Tools and Applications 36 (2008) 37–54.
[54] M. van Setten, M. Veenstra, A. Nijholt, Prediction strategies: combining prediction techniques to optimize personalization, in: Proceedings of the 2nd Workshop on Personalization in Future TV, 2002, pp. 23–32.
[55] M.G. Vozalis, K.G. Margaritis, Applying SVD on item-based filtering, in: Proceedings of the 5th International Conference on Intelligent Systems Design and Applications (ISDA'05), 2005, pp. 464–469.
[56] M.G. Vozalis, K.G. Margaritis, Applying SVD on generalized item-based filtering, Web-based Recommender Systems of the International Journal of Computer Science and Applications 3 (3) (2006) 27–51. Special Issue.
[57] M.G. Vozalis, K.G. Margaritis, Using SVD and demographic data for the enhancement of generalized collaborative filtering, Information Sciences 177 (15) (2007) 3017–3037.
[58] H. Wittig, C. Griwodz, Intelligent media agents in interactive television systems, in: Proceedings of the International Conference on Multimedia Computing and Systems (ICMCS'95), IEEE Computer Society, 1995, pp. 182–189.
[59] XMLTV Project Home Page, 2010. URL: <http://xmltv.org/wiki>.
[60] J. Xu, L. Zhang, H. Lu, Y. Li, The development and prospect of personalized TV program recommendation systems, in: Proceedings of the 4th International Symposium on Multimedia Software Engineering (MSE'02), 2002, pp. 82–89.
[61] H. Zhang, S. Zheng, Personalized TV program recommendation based on TV-anytime metadata, in: Proceedings of the Ninth International Symposium on Consumer Electronics (ISCE'05), IEEE Computer Society Press, 2005, pp. 242–246.
[62] C. Ziegler, S.M. McNee, J.A. Konstan, G. Lausen, Improving recommendation lists through topic diversification, in: Proceedings of the 14th International Conference on World Wide Web (WWW'05), ACM, 2005, pp. 22–32.