What's on TV Tonight? An Efficient and Effective Personalized Recommender System of TV Programs

Ana Belén Barragáns Martínez, *Member*, IEEE, José J. Pazos Arias, *Member*, IEEE, Ana Fernández Vilas, Jorge García Duque, and Martín López Nores

Abstract — *With the expansion of digital networks and TV devices and the rapid increase of the number of channels, people are exposed to an information overload, due to the presence of several hundreds of alternative programs to watch. In this context, personalization is achieved with the employment of algorithms and data collection schemes that predict and recommend to television viewers content that match their interests and/or needs.*

This paper introduces queveo.tv: a personalized TV program recommendation system. The proposed hybrid approach (combining content-filtering techniques with those based on collaborative filtering) also provides all typical advantages of any social network as comments, tagging, ratings, etc. This web 2.0 application has been devised to enormously simplify the task of selecting what program to watch on TV.

Index Terms — Recommender systems, Contents personalization, Collaborative filtering, SVD.

I. INTRODUCTION

With the merge of DVB-T/C/S, TV users today are excited due to the high number of channels in their TV sets; however, they are also confused how they can find their preferences from the thousands of programs. In this context, the personalized TV program recommendation systems allow us to cope with this problem by automatically matching user's wishes to TV programs and recommending the ones with high user preference. Recommender systems [1], [2] are usually classified into the following categories:

Content-based recommenders: The user will be recommended items similar to the ones the user preferred in the past. These systems (also called Information Filtering (IF) systems) require a profile of user needs.

Collaborative or social recommenders: The user will be recommended items that people with similar tastes and preferences liked in the past. In the Collaborative Filtering (CF) approach, the recommender system identifies users who share the same preferences (e.g. rating patterns) with the active user, and proposes items which the like-minded users favored (and the active user has not yet seen).

Both methods present advantages and disadvantages and significant research effort has been devoted to hybrid recommendations methods that combine collaborative and content-based filtering exploiting the advantages of both methods [3], [4], [5]. The more significant drawback of each of the above techniques is precisely solved by using the other method.

On the one hand, the problem of *overspecialization* of IF techniques (only very similar items to previous items consumed by the user are recommended) does not appear using collaborative filtering because this algorithm can recommend other items which the like-minded users flagged as of great value.

On the other hand, 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. Similarly, the *cold-start* problem is caused by new users in the system which have not submitted any ratings. Without any information about the user the system is not able to guess the user's preferences and generate recommendations until a few items have been rated.

Both problems can be solved by using IF 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.

To overcome these limitations, we propose to adopt a hybrid approach between content-based matching and collaborative filtering. This hybrid proposal is specially interesting in TV recommenders domain since TV programs recommendation has been typically realized through personalization of the Electronic Program Guide (EPG), using only content filtering techniques.

But there is still another limitation, not yet solved, of collaborative filtering. Although this method has been shown to produce high quality recommendations, its performance degrades with the number of users and programs (*sparsity* problem). Therefore, new recommender system technologies are needed that can quickly produce high quality recommendations, even for very large-scale problems. We have explored the technology called *Singular Value Decomposition* (SVD) to reduce the dimensionality of recommender system database.

This paper introduces queveo.tv: a web 2.0 application which has been devised to enormously simplify the task of selecting what program to watch at TV. This hybrid approach between content-based and collaborative filtering will be enriched with SVD technology to avoid *sparsity* problems of collaborative recommenders. It will also count with the additional users' contents usually provided in a social network.

Manuscript received December 30, 2008

The rest of this paper is organized as follows. Section II discusses previous related work. Section III provides the architecture of the system as well as a motivating example. Section IV describes the content-based filtering and social filtering strategies. The details of the SVD algorithm are provided in Section V together with our version of CF algorithm enhanced with SVD. In section VI, we present the evaluation results, whereas the prototype implementation is described in Section VII. Finally, section VIII concludes and points out directions for future work.

II. 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. So, based on how recommendations are made, recommenders systems are usually classified into the following three categories [6], [1], [7]:

1. *Content-based recommendations*: Recommendations are provided by automatically matching a customer's interests with items' contents. Items that are similar to ones the user preferred in the past are now recommended. 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.

2. *Collaborative filtering*: Recommendations are made for items that people with similar tastes and preferences liked in the past. Most CF-based algorithms recommend items for users based on the nearest-neighborhood method [8]. A typical nearest-neighborhood procedure can be separated into three steps. The first step is to determine a neighborhood for the active user or item through the similarity value, which is usually defined by the Pearson correlations. Then, a prediction is calculated from a weighted combination of selected neighbors' ratings for the active user or given item. Finally, a descending sorted list is made from these predictions. Recommendations are generated by choosing the top *N* items on this list.

   According to [9], algorithms for collaborative recommendation can be grouped into two general classes: memory-based and model-based. Memory-based algorithms [9], [10] are heuristics that make ratings predictions based on the entire collection of items previously rated by users. Model- based algorithms [11] use the collection of ratings to learn a model, which is then used to make ratings prediction. Therefore, in comparison with model-based methods, memory-based algorithms can be considered "lazy learning" methods because they do not build a model, but instead perform the heuristic computations when recommendations are requested [6]. 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.

   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 [10] and the cosine-based approach [9], [12].

3. *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 contents filtering.

   Burke [4] classifies hybridization techniques into seven classes: *weighted* where each of the recommendation approaches makes predictions which are then combined into a single prediction; *switching* where one of the recommendation techniques is selected to make the prediction when certain criteria are met; *mixed* in which predictions from each of the recommendation techniques are presented to the user; *feature combination* where a single prediction algorithm is provided with features from different recommendation techniques; *cascade* where the output of from one recommendation technique is refined by another; *feature augmentation* where the output of one recommendation technique is fed to another, and *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 produce independently from each other a prediction 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.

Collaborative recommender systems suffer from various fundamental problems that reduce the quality of the generated predictions. Such problems are *first-rater*, *cold-start* and *sparsity*. A number of solutions for these problems have been proposed. The two first are solved in our proposal by using the content-based recommender. Sparsity problem implies that, in the absence of sufficient amount of available ratings for the active user, similarities may be lead to erroneous selection of actually "bad" neighbors as good ones and vice versa.

In order to avoid the sparsity problem, we will apply in our collaborative recommender the solution proposed by [13]

which is focused on the case of singular value decomposition that 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.

III. Architecture of the System

In Fig. 1 we present an overview of our recommender framework.

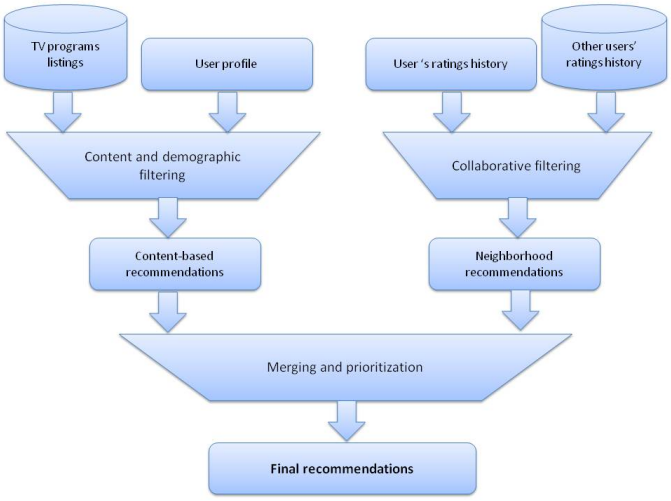

Fig. 1. Overview of our recommender framework

Our system implements collaborative filtering, content-based filtering, and a merging algorithm that combines the results provided by both methods above. The CF algorithm can be viewed as a generalized CF approach which utilizes SVD as an augmenting technique in order to enhance the recommender system's efficiency and improve the accuracy of the generated predictions. Our approach 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. Our system also uses memory-based user-to-item correlation to predict the level of interest for each item and calculate the items' neighborhood.

In first place, as we can see in Fig. 1, the system gathers all TV programming information of all the available channels where each program is tagged with suitable words describing its content. This represents a great amount of information which should be filtered according to the user preferences, for example, which kind of programs the user likes, which 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 is done either explicitly (by querying the user) and implicitly (by observing the user's behavior). Apart 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 [14], obtaining this way 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. In web-based recommender systems (like queveo.tv) with rich interaction data available, users are typically asked to rate observed items in a one to five scale (e.g. amazon.com). Moreover in queveo.tv other types of data are collected implicitly.

Returning to Fig. 1, the user ratings' history is used to find users with similar tastes, called *neighbors*. This process was initially done 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 rated-based mechanism, that is, 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.

The final stage consists, as shown in Fig. 1, in merging the recommendations suggested by both techniques, prioritizing, for example, those programs which appear in both listings or have highest ratings.

A. Motivating example

Let us suppose that Ann and Peter are two users of this social network with very similar TV tastes focused, specially, in sports (soccer, more specifically) and TV series. Once Ann is logged into the web, the system immediately asks her for feedback with respect to the last given recommendations. Ann can select the "not seen" option, or if she could actually watch any of the recommended programs, she can rate them in a 1 to 5 scale. Next, the system offers her a list of program recommendations for that day. As expected, the system will filter the programs offered that day taking into account her preferences.

Since Ann has specified in her profile that she has not access to cable TV nor pay-per-view channels, and that she is usually watching TV between 20:30 and 22:30, the content-based recommender will suggest only programs in public channels as shown in Table 1.

Table 1
Recommended Programs Using Information Filtering

Time	Channel	Program
20:30	LaSexta	Series: **Prison Break**
		Season 3, Ep. 13: *The Art of the Deal*
20:45	Cuatro	Soccer: **Netherlands – Spain**
		Semi-final EuroCup'08
22:00	Telecinco	Series: **CSI: Las Vegas**
		Season 7, Ep. 18: *Empty Eyes*

However, the final recommendations will be enriched thanks to the collaborative filtering. Since Peter and other neighbors have rated the soccer match as very interesting (it is a semifinal match where Spain is participating), the system will order the programs and prioritize the highest rated ones.

On the other hand, since Peter has rated with the maximum value the premiere of a new series (that Peter saw six months ago following the American broadcast), this recommendation will also appear even though it will be broadcast outside the preferred schedule of Ann. It should be noted that the restriction concerned to which available channels she has is always respected. However, the provided schedule to watch programs can be ignored because she may decide to record the program if it is really highly recommended.

In addition, Peter has also given good rates to a documentary about EuroCup which is scheduled outside Ann's leisure time, but it may be of interest for users who like soccer although they have not specified special interest in documentaries in their profile, the case of Ann. So, the recommendations given to Ann will be those shown in Table 2.

TABLE 2
RECOMMENDED PROGRAMS USING INFORMATION AND COLLABORATIVE FILTERING

Time	Channel	Program	Rating
20:45	Cuatro	Soccer: **Netherlands – Spain** Semi-final EuroCup'08 Recommended by your neighbor **Peter**	9
22:45	Cuatro	Series: **Dexter** Season 1, Ep. 1: *Dexter* Recommended by your neighbor **Peter**	8.5
20:30	LaSexta	Series: **Prison Break** Season 3, Ep. 13: *The Art of the Deal* This series fits your interests	7.5
15:30	TVE1	Documentary: **Spain's History in previous EuroCups** Recommended by your neighbor **Peter**	7.25
22:00	Telecinco	Series: **CSI: Las Vegas** Season 7, Ep. 18: *Empty Eyes* This series fits your interests	7

It is interesting to highlight that the user will have access to not only personalized program lists but also the recommendation reasons, which explain why the program have been recommended. This is of great value for increasing the trust in recommendations [15], [16].

Properties of IF recommenders are useful because programs not seen by her neighbors or previously rated will be presented to the user since they match her profile. The overspecialization problem is eliminated with the second filtering where the system takes into account collective information from similar users and can provide different kinds of programs or even outside the preferred schedule (as happened in the previous example with the documentary or Dexter). Thanks to the use of IF filtering the cold start and first-rater problems of CF filtering does not exist. In next sections, we will see how sparsity problem is also eliminated thanks to SVD technology.

IV. METHODOLOGY

In this section we will explain the methodologies followed to implement both types of filtering.

A. 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 [17], and nowadays, it is often used in TV program recommendation systems [16].

The basic concept of the vector space model is the selection of an item vector that it is most similar to a retrieval key when there is one retrieval key and several item vector 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 [18]. These practical systems are able to select the programs that have the same title or cast recorded by a user in the past.

Existing TV recommendation systems have as drawback that 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. In our system this does not happen, and the used collaborative filtering algorithm is explained in the following subsection.

B. 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 his taste.

In our approach we use part of both methods. In 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 N most similar users). On the other hand, our system makes use of the item-based approach 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. Similarly to user-based collaboration filtering, item-based filtering is based on the creation of neighborhoods. Yet, unlike the user-based CF approach, those neighbors consist of similar items rather than similar users.

The execution steps of the algorithm are (a) *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. (b) *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 metrics (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. (c) *Prediction Generation*, where final predictions are calculated as a weighted sum of ratings given by a user on all items included in the active item's neighborhood.

V. SVD TECHNOLOGY

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.

A. Singular Value Decomposition

Singular Value Decomposition (SVD) [19] is a matrix decomposition technique which takes an $m \times n$ matrix A, with rank r, and decomposes it as follows:

$$SVD(A) = U \times S \times V^T$$

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 $s_1 \geq s_2 \geq \ldots \geq s_r$. Accordingly, the first r columns of U are eigenvectors of AA^T and represent the left singular vectors of A, spanning the column space. The first r columns of V are eigenvectors of A^TA and represent the right singular vectors of A, 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 A. By retaining the first $k \ll r$ 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 A. 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 A_k which is defined as:

$$A_k = U_k \times S_k \times V_k^T$$

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

B. Our CF algorithm enhanced with SVD

We will now describe the steps of how SVD can be combined with item-based filtering in order to make the base algorithm more scalable [20].

1. 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. Preprocess user-item matrix R in order to eliminate all missing data values. The preprocessing is described in detail here:
   a) Compute the average of all rows, r_i, where i=1,2,…,m, and the average of all columns, r_j, where j=1,2,…,n, from the user-item matrix, R.
   b) Replace all matrix entries that have no values with the corresponding *column* average, r_j, which leads to a new filled-in matrix, $R_{filled\text{-}in}$.
   c) Subtract the corresponding *row* average, r_i, from all the slots of the new filled-in matrix, $R_{filled\text{-}in}$, and obtain the normalized matrix R_{norm}.

3. 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. 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. Compute $\sqrt{S_k}$ and then calculate two matrix products: $U_k \times \sqrt{S_k}^T$, which represents m users, 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. 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:

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

   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. Conclude with prediction generation, achieved by the following weighted sum:

$$pr_{aj} = \frac{\sum_{k=1}^{l} sim_{jk} * (rr_{ak} + \overline{r}_a)}{\sum_{k=1}^{l} \left| sim_{jk} \right|}$$

   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.

C. Benefits of application

As explained in [20], a recommender system running item-based filtering with a lower dimensional representation, as described in the previous, 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 $k<<m$, 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 the following paragraphs.

VI. 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 [21]. Nevertheless, Herlocker *et al.* [22] 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 [22] into three categories: (1) *predictive accuracy metrics*, which measure how close the recommender's predictions are to the true user ratings; (2) *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/him, and; (3) *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.

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. MAE is measured only for those items for which a user has expressed her/his 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) – 5 (excellent) numerical scale. Starting from the initial data set, five distinct splits of training and test data were generated.

The aim of the experiments which will be described in the following sections is first to locate the optimal parameter settings for the item-based enhanced by SVD filtering algorithm.

A. 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 used, 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. 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 [20]), 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}, a bigger set of values than in [20] where authors used only eight values of k. Fig. 2 collects the Mean Absolute Errors observed by those runs, averaged over the five data splits from the data set.

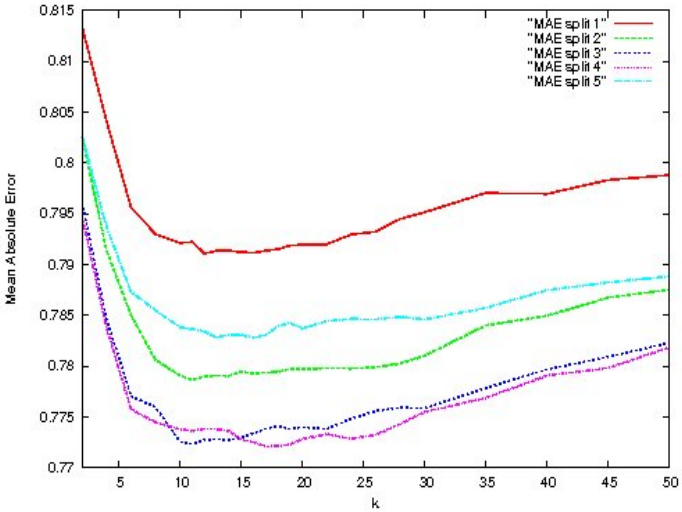

Fig. 2. MAE values for the five sets of data

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 [20] where authors obtained as optimal k = 6.

B. Future work in evaluation

Besides prediction accuracy our approach is also concerned with the recommendation coverage (the number of programs that prediction can be made) 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 [23]: 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).

On the other hand, in traditional information retrieval domain, system's quality is evaluated by *recall* and *precision*. For TV program recommendation systems, their quality can be appraised but these two parameters also, but some modification to them is needed, as explained in [24]. 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:

$$recall = \frac{|watched \cap recommended|}{|watched|}$$

$$precision = \frac{|watched \cap recommended|}{|recommended|}$$

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 the combination of the two, giving equal weight to each, the system's quality can be evaluated by *F1*:

$$F1 = \frac{2 \times recall \times precision}{recall + precision}$$

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

VII. Implementation Details

The development of queveo.tv is completely 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, programs, and TV channels are implemented in *MySQL*. For gathering the complete listings of all TV channels, the application makes use of XMLTV [25] which is a set of programs to process TV listings, storing them in an XMLTV format (based on XML). There are backends to download TV listings, filter programs and *Perl* libraries to process listings.

In order to compute the singular value decomposition, we use *linalg* [26] that is a fast, LAPACK-based library for real and complex matrices. To add social network features and to improve the web presentation (not yet finished), queveo.tv makes use of *tog* [27].

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. We are currently validating this version with undergraduate students and we plan to extend the functionalities adding the possibility of creating series' blogs where fans can comment their favorite episodes, users' groups focused in preferred programs or particular tastes as terror films, 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. queveo.tv will provide and ad hoc social community including viewers watching the same show on TV.

For not-registered 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 (or simply hit a button for the default "what's on now"), sort through the resulting list and select programs to watch. queveo.tv shows the suggested TV schedule for "today", "tomorrow" and "in 2 days" because most of users only plan a TV schedule for the following day or they did not plan at all. Many users only use a guide to determine what is currently on TV [28]. queveo.tv also allows users who serve as self-proclaimed editors to give tips that contain recommendations volunteered by these so-called opinion leaders.

VIII. Conclusions and Future Work

The proposed hybrid system combines content-filtering techniques with those based on collaborative filtering. Apart from eliminating drawbacks of each kind of system, it provides all typical advantages of any social network as comments, tagging, ratings, etc. 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 stored offline component. 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 this 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 want to test our application with potential users in order to get an appraisal of our personalization techniques. Empirical evaluation of personalization methods is crucial for their validity, although often neglected by researchers. We are also preparing a questionnaire to be filled by the users in order to make a correlation between the system's proposal and the user's opinion.

In a social network, developers always face the problem of motivating the users to contribute [29]. Future systems will likely need to offer some incentive for the provision of recommendations by making it a prerequisite for receiving recommendations or by offering monetary compensation. The users' feedback is even more important in a TV recommender and the reason is derived from the characteristics of TV programs: except on rebroadcast ones, they are non-persistent items so the feedback should be provided as soon as possible. In [27] authors investigate the use of non-monetary incentives in online communities. They are specially interested in two common design features which facilitate social comparison: leaderboards and contribution-based status levels.

There is a fundamental characteristic in users' attitudes between entering a community platform that should also be taken into account. When interacting with a non-commercial collaborative filtering system, users know that they have to provide ratings in order to help build up community knowledge and to receive good recommendations themselves in return. So what they give and what they receive are both of non-monetary value [28]. We propose to make use of recommendations of opinion leaders. It divides users into two distinct groups: a large majority of consumers that contribute only a little to a mass-average and a group of self-selected leaders that are willing to spend more effort on the system for getting social or monetary reward.

Finally, the growing presence of mobile devices such as PDAs and smartphones, along with advances in wireless network communication technologies, have created new opportunities for making the applications and services available on these hosting devices more intelligent and supportive to the user. So we also plan to extend the service to be offered to handheld devices and interactive digital TV. 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 PDA, cellular phone or set-top-box with internet access.

Acknowledgment

Thanks should be given to Alberto Lago and Óscar Álvarez, two undergraduate students which have contributed to the implementation of the application.

References

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

[2] P. Resnick and H. R. Varian, "Recommender systems," *Communications of the ACM*, 40(3):56–58, 1997.

[3] C. Basu, H. Hirsh and W. Cohen, "Recommendation as classification: using social and content-based information in recommendation," In Proceedings of the 15th National Conference on Artificial Intelligence (AAAI'98), pages 714–720, 1998.

[4] R. Burke, "Hybrid recommender systems: survey and experiments," *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.

[5] B. Smyth and P. Cotter, "A personalized television listing service," *Communications of the ACM*, 43(8):107-111, 2000.
[6] G. Adomavicius, R. Sankaranarayanan, S. Sen and A. Tuzhilin, "Incorporating contextual information in recommender systems using a multidimensional approach," *ACM Transactions on Information Systems*, 23(1):103–145, 2005.
[7] M. Balabanovic and Y. Shoham, "Fab: content-based, collaborative recommendation," *Communications of the ACM*, 40(3):66–72, 1997.
[8] J. L. Herlocker, J. A. Konstan, A. Borchers and 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, pages 230–237, New York, NY, USA, 1999. ACM Press.
[9] J. S. Breese, D. Heckerman and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," In Proceedings of 14th Conference on Uncertainty in Artificial Intelligence, pages 43–52, 1998.
[10] P. Resnick, N. Iacovou, M. Suchack, P. Bergstrom and J. Riedl, "GroupLens: an open architecture for collaborative filtering of netnews," In Proceedings of the ACM Conference on Computer Supported Cooperative Work, pages 175–186, 1994.
[11] K. Goldberg, T. Roeder, D. Gupta and C. Perkins, "Eigentaste: a constant time collaborative filtering algorithm," *Information Retrieval*, 4(2):133–151, 2001.
[12] B. M. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Application of dimensionality reduction in recommender system – A case study," In Proc. of the Workshop on Web Mining for E–Commerce – Challenges and Opportunities (WEBKDD'00), Boston, USA, August 20, 2000.
[13] M. G. Vozalis and K. G. Margaritis, "Identifying the effects of SVD and demographic data use on generalized collaborative filtering". *International Journal of Computer Mathematics*, 85(12):1741–1763, 2008.
[14] G. Lekakos and G. M. Giaglis, "Improving the prediction accuracy of recommendation algorithms: Approaches anchored on human factors," *Interacting with Computers*, 18(3):410–431, 2006.
[15] J. L. Herlocker, J. A. Konstan and J. Riedl, "Explaining collaborative filtering recommendations," In Proceedings of the ACM conference on Computer Supported Cooperative Work, pages 241–250, 2000.
[16] T. Tsunoda and M. Hoshino, "Automatic metadata expansion and indirect collaborative filtering for TV program recommendation system," *Multimedia Tools Appl*., 36:37–54, 2008.
[17] D. L. Lee, H. Chuang and K. E. Seamons, "Document ranking and the vector-space model," *IEEE Software*, 14(2):67–75, 1997.
[18] TiVo, Inc. (2002) TiVoTM homepage from http://www.tivo.com/.
[19] G. Golub and C. V. Loan, "Matrix computations," Johns Hopkins Studies in Mathematical Sciences, Baltimore, 3rd edition, 1996.
[20] M. G. Vozalis and K. G. Margaritis, "Applying SVD on item-based filtering," In Proceedings of 5th International Conference on Intelligent Systems Design and Applications (ISDA'05), pages 464–469, 2005.
[21] J. Herlocker, J. A. Konstan and J. Riedl, "An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms," *Information Retrieval*, 5(4):287–310, 2002.
[22] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Transactions on Information Systems*, 22(1):5–53, 2004.
[23] M. van Setten, M. Veenstra and A. Nijholt, "Prediction strategies: combining prediction techniques to optimize personalization," In Proceedings of the 2nd Workshop on Personalization in Future TV, pages 23–32, 2002.
[24] J. Xu, L. Zhang, H. Lu and Y. Li, "The development and prospect of personalized TV program recommendation systems," In Proceedings of 4th International Symposium on Multimedia Software Engineering (MSE'02), pages 82–89, 2002.
[25] XMLTV project home page. http://xmltv.org/wiki.
[26] Linalg project home page: http://rubyforge.org/projects/linalg/.
[27] Tog project home page: http://www.toghq.com/
[28] P. Baudisch, "Dynamic Information Filtering," Ph.D. Thesis. GMD Research Series, 2001, No. 16. GMD Forschungszentrum Informationstechnik GmbH, Sankt Augustin.
[29] A. M. Rashid, K. Ling, R. D. Tassone, P. Resnick, R. Kraut, and J. Riedl, "Motivating participation by displaying the value of contribution," In Proc. of the SIGCHI Conference on Human Factors in Computing Systems (CHI'06), pages 955–958, New York, NY, USA, 2006. ACM Press.
[30] M. Harper, S. X. Li, Y. Chen and J. A. Konstan, "Social comparisons to motivate contributions to an online community," In Second International Conference on Persuasive Technology (PERSUASIVE'07), pages 148–159, 2007.
[31] M. Zanker and M. Jessenitschnig, "Case-studies on exploiting explicit customer requirements in recommender systems," *User Modeling and User-Adapted Interaction: The Journal of Personalization Research*, A. Tuzhilin and B. Mobasher (Eds.): Special issue on Data Mining for Personalization, 19(1), 2009.

Ana Belén Barragáns Martínez is an assistant professor in the Department of Telematics Engineering of the University of Vigo since 2000, where she received her PhD in Computer Science in 2007, in the field of Software Engineering. She is a member of the Interactive Digital TV Lab, where she is specially interested in recommendation of TV programs making use of social networks and web 2.0 technologies as well as personalization of advertising. She is also member of IEEE, ACM and ACM SIGSOFT from 2003.

José Juan Pazos Arias received his degree in Telecommunications Engineering from the Polytechnic University of Madrid (Spain-UPM) in 1987, and his Ph.D. degree in Computer Science from the Department of Telematics Engineering at the same University in 1995. He is the director of the Networking and Software Engineering Group in the University of Vigo, which is currently involved with projects on middleware and applications for Interactive Digital TV.

Ana Fernández Vilas received her Ph.D. degree in Computer Science from the University of Vigo in 2002. In 1997, she joined the Department of Telematics Engineering (University of Vigo) where she is currently an associate professor. She is engaged in web services technologies and ubiquitous computing environments, being a member of the Interactive Digital TV Lab.

Jorge García Duque is an associate professor in the Department of Telematics Engineering at the University of Vigo. He received the Ph.D. degree in Computer Science from the University of Vigo in 2000. His major research interests are related to the development of new software methodologies and services for Interactive Digital TV.

Martín López Nores is an assistant professor in the Department of Telematics Engineering at the University of Vigo. He received the Ph.D. degree in Computer Science from this university in 2006. His research deals primarily with the design of personalization architectures for a range of DTV applications, considering both fixed and mobile receivers.