International Journal of Computer Science & Applications
Vol. 3 Issue 3, pp 27- 51

Applying SVD on Generalized Item-based Filtering

Manolis G. Vozalis[1] and Konstantinos G. Margaritis

Parallel Distributed Processing Laboratory,
Department of Applied Informatics, University of Macedonia,
Egnatia 156, P.O. 1591, 54006, Thessaloniki, Greece

Abstract

In this paper we examine the use of a matrix factorization technique called Singular Value Decomposition (SVD) along with demographic information in Item-Based Collaborative Filtering. After a brief introduction to SVD and to some of its previous applications in Recommender Systems, we proceed with the presentation of two distinct but related algorithms. The first algorithm uses SVD in order to reduce the dimension of the active item's neighborhood. The second algorithm initially enhances Item-based Filtering with demographic information and then applies SVD at various points of the filtering procedure. The presentations of both algorithms include a detailed step-by-step description and a series of experiments. In both cases the results show that a reduction in the dimension of the item neighborhood via SVD, either by itself or combined with the usage of relevant demographic information, is promising, since it does not only tackle some of the recorded problems of Recommender Systems, but also assists in increasing the accuracy of systems employing it.

1 Introduction

Recommender Systems were introduced as a computer-based intelligent technique that assists people with the problem of information and product overload, their aim being to provide efficient personalized solutions in e-business domains, that would benefit both the customer and the merchant. They feature two basic entities: the *user* and the *item*. A user who utilizes the Recommender System, provides his opinion about past items. The purpose of the Recommender System is to generate suggestions about new items for that particular user. The process is based on the input provided, which is usually expressed in the form of ratings from that user, and the filtering algorithm, which is applied on that input.

[1] To whom correspondence should be addressed, e-mail: mans@uom.gr

Recommender Systems suffer from a number of fundamental problems that cause a reduction in the quality of the generated predictions, such as *sparsity*, *scalability*, and *synonymy*. A number of solutions have been introduced, intending to solve those problems [1, 2, 3]. We will focus on the case of Singular Value Decomposition which only recently was proposed by various Recommender Systems researchers as a possible means of alleviating the aforementioned problems. The first algorithm we present in this paper utilizes SVD as a technique that is able to effectively reduce the dimension of the user-item data matrix, and then it executes Item-based Filtering with this low rank representation to generate its predictions. The second algorithm takes things a step further: It executes SVD to achieve similar results with the first approach, but at the same time, it enhances the filtering procedure by taking into account relevant demographic data as a source of additional information.

The subsequent sections are structured as follows: We first introduce the concept of Singular Value Decomposition. Then, we present some Recommender Systems which have employed SVD in order to improve their performance. We include a brief discussion of the collaborative filtering algorithm which was utilized to support our technique, and provide an analysis of our experimental methodology, regarding the data set and evaluation metrics. At this point we give a step-by-step description of our first proposed filtering method. The experimental work which follows, closes by comparing our technique with the base filtering algorithm. A similar presentation is included for the second filtering method we propose: A detailed description of the algorithm is followed by an extended series of related experiments. We conclude with a summary and possible directions for future work.

2 Singular Value Decomposition (SVD)

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

$$SVD(A) = U \times S \times V^T \tag{1}$$

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 $s_1 \geq s_2 \geq \cdots \geq s_r$. 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 $A^T A$ 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 $k << r$ 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 matrix 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:

$$A_k = U_k \times S_k \times V_k^T \tag{2}$$

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.

3 Using SVD in Recommender Systems

SVD, as part of *Latent Semantic Indexing (LSI)*, was widely used in the area of Information Retrieval [4] in order to solve the problems of *synonymy*, which is a result of the many possible ways to express a known concept, and *polysemy*, which comes from the fact that most words usually have multiple meanings. Furthermore, techniques like *SVD-updating* or *folding-in* were proposed to alleviate the problem of *updating*, which refers to the process of new terms and/or documents being added to existing matrices [5].

Those ideas were adopted by researchers in the area of Recommender Systems. Initially, Billsus and Pazzani [6] took advantage of the properties of SVD in their attempts to formulate collaborative filtering as a classification problem. In their work they utilize SVD as a dimensionality reduction technique, before they feed their data matrix, which is now represented in the reduced feature space, into an Artificial Neural Network (ANN). This ANN, or, as the authors claim, any alternative Machine Learning algorithm, can then be trained in order to generate predictions.

GroupLens have also applied SVD in at least 3 distinct cases regarding Recommender Systems: (i) in an approach that reduces the dimensionality of the user-item space and forms predictions in that reduced space, by not building explicit neighborhoods at any point of the procedure, (ii) in an approach that generates a user-neighborhood in the SVD reduced space and then applies normal user-based collaborative filtering on it, and (iii) in an approach that aims at increasing the scalability by applying folding-in for the incremental computation of the user-item model [7, 8].

Finally, Goldberg et al. use Principal Component Analysis [9], a technique very similar to SVD, in order to optimally project highly correlated data along a smaller number of orthogonal dimensions. Then, their Eigentaste algorithm clusters the projected data, being concluded by an online computation of recommendations.

4 The Base Algorithm: Item-based Filtering

In this section we will briefly discuss the filtering algorithm which will be utilized by the proposed technique. Similarly to User-based Collaborative Filtering, Item-

based Filtering is based on the creation of neighborhoods. Yet, unlike the User-based Collaborative Filtering approach, those neighbors consist of similar items rather than similar users [10].

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. 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.

5 Experimental Methodology

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.

Several techniques have been used to evaluate Recommender Systems. Those techniques have been divided by Herlocker et al. [11] into three categories: *Predictive Accuracy Metrics*, *Classification Accuracy Metrics*, and *Rank Accuracy Metrics*. The choice among those 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 proceeded in the selection of the initial evaluation metric for our experiments. That metric was *Mean Absolute Error (MAE)*. It is a statistical accuracy metric which 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 his opinion.

6 Algorithm 1: Item-based Filtering Enhanced by SVD

6.1 Description

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.

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, denoted by $\perp$, 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}.

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: $R_{norm} = U \cdot S \cdot V^T$.

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 $R_{red} = U_k \cdot S_k \cdot V_k^T$, 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: $U_k \cdot \sqrt{S_k}^T$, which represents m users and $\sqrt{S_k} \cdot V_k^T$, 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:

$$sim_{jf} = adjcorr_{jf} = \frac{\sum_{i=1}^{k} mr_{ij} \cdot mr_{if}}{\sqrt{\sum_{i=1}^{k} {mr_{ij}}^2 \sum_{i=1}^{k} {mr_{if}}^2}} \tag{3}$$

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:

$$pr_{aj} = \frac{\sum_{k=1}^{l} sim_{jk} * (rr_{ak} + \bar{r_a})}{\sum_{k=1}^{l} |sim_{jk}|} \tag{4}$$

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, $\bar{r_a}$, back since it was subtracted during the normalization step of the preprocessing.

6.2 Benefits of Application

A Recommender System running Item-based Filtering with a lower dimensional representation, as described in the previous section, 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 from 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.

6.3 Experiments

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. Once those settings are found and applied, we will compare the results with the predictions generated by a plain Item-based Recommender System running also with optimal settings.

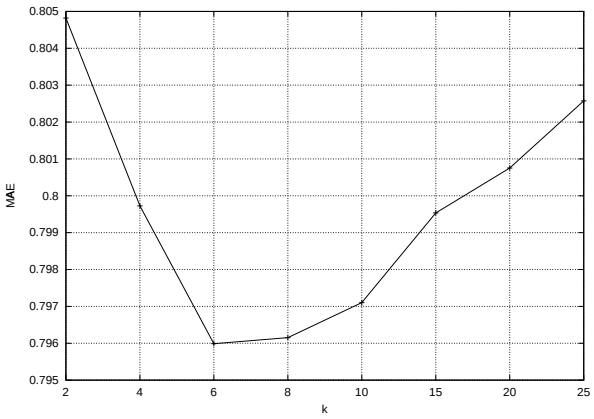

Figure 1. Identifying the best value of k

6.3.1 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 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, and ran our algorithm repeatedly for different values of k, $k = \{2, 4, 6, 8, 10, 15, 20, 25\}$. Figure 1 collects the Mean Absolute Errors observed by those runs, averaged over the 5 data splits from the data set.

Our results indicate that applying Item-based Filtering on a neighborhood of reduced dimension displays a similar behavior to utilizing a low dimensional representation in User-based Filtering. The quality of our Recommender System's predictions follows the pattern reported for the ML data set in Sarwar [7]. Specifically, at first, the predictions' accuracy improves as we increase the rank of the lower dimension space and quickly reaches its peak for $k = 6$. 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. Consequently, in our subsequent experiments we will utilize a fixed value of $k = 6$.

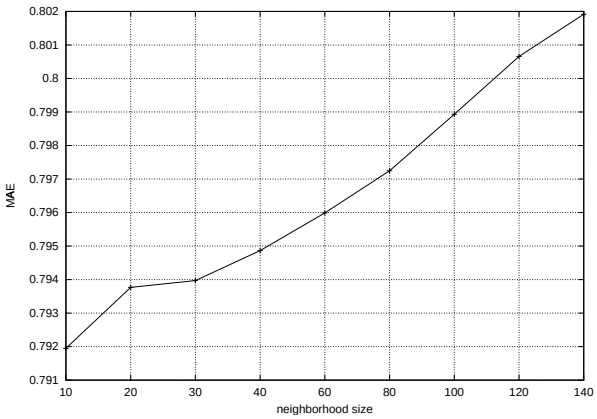

Figure 2. Identifying the best neighborhood size

6.3.2 Locating the optimal value for the neighborhood size

As reported in previous work [12], the size of the neighborhood greatly affects the behavior of Collaborative Filtering algorithms. It is a common trend for such algorithms to show an initial improvement in accuracy, as the neighborhood size increases, reach their maximum performance for a specific threshold, and remain stable or show a slight decline after that threshold.

For our experiments, we kept the dimension of the lower rank representation fixed to 6, retaining the ideal value for k as recorded in the preceding section, and varied the size of the active item's neighborhood, *neigh-size*=$\{$*10-140*$\}$. The errors observed from the runs of our algorithm under those settings, averaged over the 5 data splits of the data set, are collected in Figure 2.

Other researchers have reported that the ideal number of neighbors in a collaborative filtering algorithm is data set dependent [7]. Furthermore, experimental results, involving neighborhoods of varying sizes [12], have indicated that Item-based Filtering reaches its peak performance for quite smaller neighborhoods than those required for similar experiments in User-based Filtering. In that sense, the behavior recorded in Figure 2 is quite interesting. The accuracy of Item-based Filtering, with its original user-item matrix reduced by SVD, starts with a rather low error value for a neighborhood including 10 items, and gets steadily worse for neighborhoods whose size continuously increases, concluding with a maximum size of 140 items. As a result, according to the findings of this experiment we can assume that a neighborhood including only 10 items, which are the most similar to the active item, is able to generate the most accurate predictions.

6.3.3 Comparing Item-based Filtering enhanced by SVD with plain Item-based Filtering

Once the optimal settings for our algorithm were located, we were able to proceed with our final experiment. The purpose of this experiment was to evaluate the accuracy of Item-based Filtering, when utilizing a low-dimensional neighborhood, by

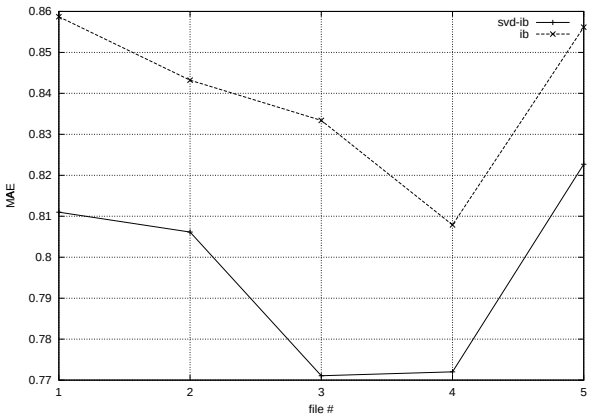

Figure 3. Comparing Item-based Filtering enhanced by SVD with plain Item-based Filtering

Table 1. Average MAEs for both neighborhood dimensions

	high-dimensional Item-based	**low-dimensional Item-based**
MAE	0.83987391	0.79658809

contrasting it with Item-based Filtering employing the original, high-dimensional neighborhood. The parameters in both implementations were set for optimal predictions.

Figure 3 includes the Mean Absolute Errors for high (*ib*) and low (*svd-ib*) dimensions, as observed for each of the 5 data splits of the data set. These error values are then averaged and Table 1 records the final results for both implementations.

From both the preceding figure and table, we can conclude that applying Item-based Filtering on the low-rank neighborhood, provides a clear improvement over the higher dimension neighborhood. The interesting part is that this improvement, owed to the low-rank representation, is not limited only to the solution of the sparsity problem, or to the improvement of the scalability of the Recommender System. This improvement can also be measured by a considerable increase in the values of the achieved accuracy.

7 Algorithm 2: Applying SVD on demographically enhanced Item-based Filtering

Our second algorithm, which we call *IdemSvd*, picks Item-based Filtering as its starting point, while taking advantage of SVD and item demographic information during its execution. The sections that follow provide a description of 4 distinct implementations of IdemSvd (*I-Demog*, *I-Rsvd*, *I-Dsvd* and *I-2svd*). Each implementation incorporates a different level of SVD and demographic data involvement

in the filtering procedure. Next comes a detailed experimental section. The efficiency of the implementations is tested and contrasted, not only against each other, but, also, against the selected base algorithm, Item-based CF.

7.1 Description

We will now present the general steps of how SVD and demographic data can be incorporated in multiple points of Item-based Filtering in order to enhance it.

- *Step 1a*: Construct demographic vectors for the m users and n items that participate in the recommendation process. The information required for those vectors can be usually found in the utilized collaborative filtering data sets, like MovieLens and EachMovie. Once constructed, the demographic vectors are collected in array D_i.
- *Step 1b*: At this point we have to select one of two possible scenarios:
  - *Case 1*: Leave the array of demographic vectors, D_i, intact (cases of implementations *I-Demog* and *I-Rsvd*), or,
  - *Case 2*: Perform SVD on the array of demographic vectors. (cases of implementations *I-Dsvd* and *I-2svd*)
    After applying SVD on item demographic array D_i, where $D_i = Ud_i \cdot Sd_i \cdot Vd_i^T$, perform dimensionality reduction by keeping only the k biggest singular values of Sd_i. This will lead to a "reduced" item demographic array $D_{i,k}$, where $D_{i,k} = Ud_{i,k} \cdot Sd_{i,k} \cdot Vd_{i,k}^T$. Consequently, $D_{i,k}$ collects the n "reduced" item demographic vectors, each of which is composed by k *pseudo*-features. Meanwhile, the matrix product $Ud_{i,k} \cdot \sqrt{Sd_{i,k}}^T$ represents items with the help of those k-rank pseudo-features.
- *Step 2*: Proceed with Data Representation which concludes with the construction of the initial user-item matrix, R, of size $m \times n$.
- *Step 3*: Resume with the Neighborhood Formation. Once again, we have to select one of the following possible ways:
  - *Case 1*: Neighborhood Formation *without* SVD (cases of implementations *I-Demog* and *I-Dsvd*).
    Specifically, the similarity between the active item, i_j, and a random item, i_f, can be evaluated by utilizing the correlation metric of choice, which in the case of Item-based Filtering is the Adjusted Cosine Similarity:

$$sim_{jf} = adjcorr_{jf} = \frac{\sum_{i=1}^{l} (r_{ij} - \bar{r_i})(r_{if} - \bar{r_i})}{\sqrt{\sum_{i=1}^{l} (r_{ij} - \bar{r_i})^2 \sum_{i=1}^{l} (r_{if} - \bar{r_i})^2}} \quad (5)$$

    The ratings, r_{ij} and r_{if}, which appear in eq. 5 come from the original user-item matrix, R. Also, since no pre-processing regarding R has taken place, we are required to subtract the mean user rating, $\bar{r_i}$, to atone for the different rating behaviors.

Neighborhood formation for the active item concludes by a selection of the h items most similar to i_j, according to their correlation values.

– *Case 2*: Neighborhood Formation *with* SVD. (cases of implementations *I-Rsvd* and *I-2svd*).

Specifically, pre-process array R in order to generate R_{norm}, which includes no empty slots. Apply SVD on R_{norm}, where $R_{norm} = U \cdot S \cdot V^T$, and perform the dimensionality reduction step, by keeping the l largest singular values of S. This will generate a "reduced" user-item matrix, R_{red}, where $R_{red} = U_l \cdot S_l \cdot V_l^T$. At the same time, the matrix product $\sqrt{S_l} \cdot V_l^T$ represents the n items in the l-dimensional space.

rr_{ij} denotes the rating of user u_i on item i_j, as included in the reduced matrix, R_{red}, while $\sqrt{S_l} \cdot V_l^T$ features the "meta" ratings, mr_{ij}, assigned by the l *pseudo*-users on the n items.

At this point, the similarity between the active item, i_j, and a random item, i_f, can be evaluated by calculating their Adjusted Cosine Similarity:

$$sim_{jf} = adjcorr_{jf} = \frac{\sum_{i=1}^{l} mr_{ij} \cdot mr_{if}}{\sqrt{\sum_{i=1}^{l} {mr_{ij}}^2 \sum_{i=1}^{l} {mr_{if}}^2}} \tag{6}$$

One can note that there are two important differences from plain Adjusted Cosine Similarity: (i) we are utilizing the "meta" ratings of the l *pseudo*-users on items i_j and i_f, and, (ii) we do not need to subtract the mean user ratings in order to offset the different rating behaviors of the users, since that normalization was done in the pre-processing of the user-item matrix.

Neighborhood formation for the active item concludes by a selection of the h items most similar to i_j, according to their correlation values.

- *Step 4*: Calculate the Demographic Correlation between the active item, i_a, and each of the members of its neighborhood, i_i, by computing their corresponding vector similarities:

$$dem\_cor_{ai} = vect\_sim(\overrightarrow{i_a}, \overrightarrow{i_i}) = \frac{\overrightarrow{i_a} \cdot \overrightarrow{i_i}}{||\overrightarrow{i_a}||_2 * ||\overrightarrow{i_i}||_2} \tag{7}$$

Depending on the choices made in Steps *1b* and *3*, there exist 4 possible implementations of the general algorithm, which are summarized in Table 2. They differ in whether the selected pairs of items were taken from the *reduced* or the *original* user-item matrix, and in whether *reduced* or *original* demographic vectors were utilized for the calculations of the demographic correlations.

- *Step 5*: Calculate the Enhanced Correlation, enh_cor_{ai}, for every pair of the form $\{i_a, i_i\}$, where i_a is the active item and i_i is a member of its neighborhood.

$$\begin{aligned} enh\_cor_{ai} = \alpha * rat\_cor_{ai} + \beta * dem\_cor_{ai} \\ + \gamma * (rat\_cor_{ai} * dem\_cor_{ai}) \end{aligned} \tag{8}$$

Table 2. Different levels of SVD application in Demographic Correlation calculations

Case	SVD on demographic array D	SVD on user-item matrix R
I-Demog	no	no
I-Dsvd	yes	no
I-Rsvd	no	yes
I-2svd	yes	yes

rat_cor_{ai} and dem_cor_{ai} represent the ratings-based and the demographic correlation between active item i_a and neighborhood member i_i, while α, β and γ are flags that define the participation of each of the three components.

- *Step 6*: Proceed with the final step of the recommendation procedure, which is Prediction Generation.

  In order to predict the rating of user u_q on active item i_a, we proceed with one of the following equations. The first equation is selected for the implementations when no SVD was applied on the user-item matrix R (*I-Demog*, *I-Dsvd*).

$$idem\_dsvd\_pr_{qa} = \frac{\sum_{k=1}^{h} enh\_cor_{ak} * r_{qk}}{\sum_{k=1}^{h} |enh\_cor_{ak}|} \tag{9}$$

  We note that in this equation, the utilized ratings, r_{qk} are taken from the original user-item matrix, R.

  A second equation is selected for the implementations where SVD was applied on the user-item matrix R (*I-Rsvd*, *I-2svd*).

$$idem\_2rsvd\_pr_{qa} = \frac{\sum_{k=1}^{h} enh\_cor_{ak} * (rr_{qk} + \bar{r_q})}{\sum_{k=1}^{l} |enh\_cor_{ak}|} \tag{10}$$

  We note that for this formula, we have utilized the user ratings, rr_{qk}, from the reduced user-item matrix, R_{red}. We also had to add back the original mean user rating, $\bar{r_q}$, which was subtracted in the pre-processing of R.

  Both approaches differ slightly from the one adopted by classic Collaborative Filtering algorithms in that they replace the ratings-based correlation, rat_cor_{ak}, between the active item, i_a, and any of the members of its neighborhood, i_k, by their enhanced correlation, enh_cor_{ak}. It is obvious that the enhanced correlation, as computed in the previous step, possibly incorporates the effects of SVD on the demographic vectors, via the demographic correlations, and/or on the user-item matrix, via the ratings-based correlations.

Table 3. Brief Description of I-Demog Implementations

	flags	**enhanced correlation**
item-based	$\alpha = 1$, $\beta = 0$, $\gamma = 0$	$enh\_cor = adj\_cor$
I-Demog1	$\alpha = 0$, $\beta = 0$, $\gamma = 1$	$enh\_cor = adj\_cor * dem\_cor$
I-Demog2	$\alpha = 1$, $\beta = 0$, $\gamma = 1$	$enh\_cor = adj\_cor + adj\_cor * dem\_cor$
I-Demog3	$\alpha = 1$, $\beta = 1$, $\gamma = 1$	$enh\_cor = adj\_cor + dem\_cor + adj\_cor * dem\_cor$
I-Demog4	$\alpha = 1$, $\beta = 1$, $\gamma = 0$	$enh\_cor = adj\_cor + dem\_cor$

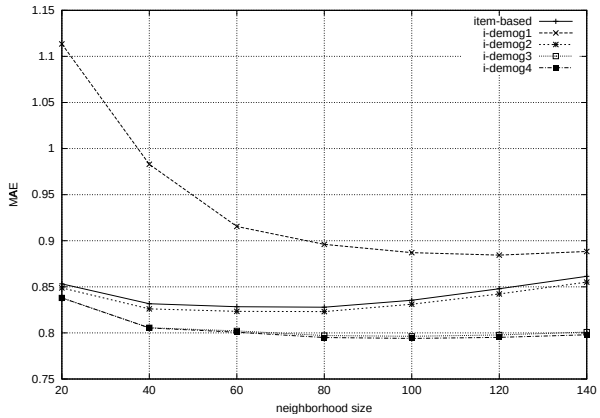

Figure 4. Comparing different implementations of I-Demog with plain Item-based Filtering

7.2 Experiments

7.2.1 Experiments with I-Demog: Demographically enhanced Item-based filtering without SVD

I-Demog can be described as an attempt to enhance the performance of Item-based Filtering, by taking advantage of demographic information regarding items. Such demographic information can be found in widely used collaborative filtering data sets. It is also present in the GroupLens data set, which we selected for our experiments.

For the experiments that follow, a number of distinct implementations of I-Demog have been tested. These implementations, distinguished by the values of the enhanced correlation flags, along with the resulting enhanced correlation equations, are gathered in Table 3. As we can see, plain Item-based filtering can be viewed as a sub-case of I-Demog, resulting from it after the assignment of the appropriate values to the flags: $\alpha = 1$, $\beta = 0$ and $\gamma = 0$.

Figure 4 compares the Mean Absolute Errors (MAE) collected from our I-Demog implementations (*i-demog1&2&3&4*) and Item-based Collaborative Filtering (*item-based*), for neighborhoods with varying sizes. Based on this figure, I-Demog3 and I-Demog4 display the best overall accuracy, clearly outperforming not only the rest of the demographically enhanced I-Demog implementations but, most importantly, plain Item-based Filtering. We pick I-Demog4, which sets

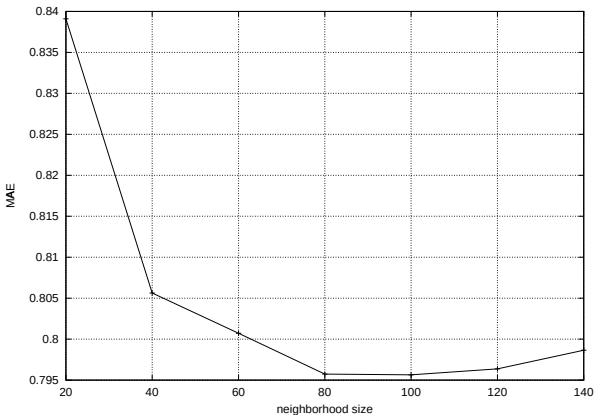

Figure 5. IdemSvd-Dsvd: Identifying the best neighborhood size

$\alpha = 1$, $\beta = 1$ and $\gamma = 0$ in its enhanced correlation equation (eq. 8), as the single best I-Demog implementation. Therefore, this particular combination of enhanced correlation flags will be utilized in the subsequent experiments, which intend to test whether I-Demog can be enhanced by the application of SVD at various points of the filtering procedure.

7.2.2 Experiments with IdemSvd-Dsvd: Applying SVD only on the demographic matrix

IdemSvd-Dsvd takes I-Demog and tests its behavior once SVD is applied on the matrix of the demographic vectors. The following experiments intend to, first, identify the optimal settings regarding the size of the neighborhood and the value of k, and then to compare our approach, optimally set, with the best predictions of plain item-based filtering.

Identifying the optimal neighborhood size for IdemSvd-Dsvd

It has been shown [10] that the size of the item neighborhood plays an important role in the recommendation procedure. The purpose of our first experiment was to locate the optimal neighborhood size to be utilized in subsequent experiments. To achieve that, we kept the value of k fixed to 6, where k corresponds to the number of dimensions retained for the demographic vectors, while setting the enhanced correlation flags, from eq. 8, to the values which yielded the most accurate predictions, according to reported experiments [13]: α=1, β=1, γ=0. At the same time, we varied the size of the neighborhood, *neigh-size*={*20-140*}. Figure 5 depicts the generated Mean Absolute Errors, averaged over all 5 data splits.

From this figure we can observe that after an initially rapid and afterwards smoother improvement, the accuracy reaches its optimal value for a neighborhood of 80 items. For sizes bigger than that, the accuracy remains, in average, mostly unchanged or slightly worse. As a result, the rest of our experiments were executed with a neighborhood set to include 80 items.

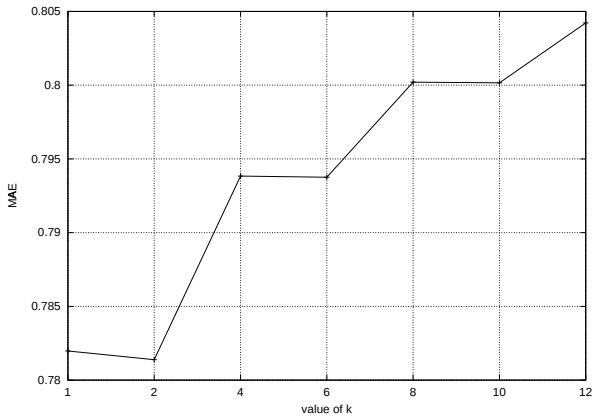

Figure 6. IdemSvd-Dsvd: Identifying the best value of k

Identifying the optimal value of k for IdemSvd-Dsvd

The second experiment's purpose was to identify the best value of k, which corresponds to the number of *pseudo*-features to be retained by the item demographic vectors after the application of SVD. As a result, we set the neighborhood size to the optimal values found by the previous experiment, assigned the appropriate values to the enhanced correlation flags (α=1, β=1, γ=0), and varied only the values of k, k={1,2,4,6,8,10,12}. The generated Mean Absolute Errors, averaged over the 5 data splits, are displayed in Figure 6.

Without forgetting that the original item demographic vectors include 18 distinct features, expressing the genres of the corresponding movies, the behavior of the line in Figure 6 can be considered as rather surprising: it shows that, on average, only 2 demographic features can describe the item in mind adequately. This number is significantly lower than the 18 original features. Furthermore, there was a single data split which yielded its best accuracy for k=1.

Based on these observations, we can assume that the demographic features, included in the utilized data set, provide information about the items which appear to be quite similar or, even, overlapping. Consequently, and by taking into account the resulting improvement in prediction accuracy, their merging should be recommended.

Comparing IdemSvd-Dsvd with plain Item-based Filtering

Having identified the optimal parameter settings for k and the size of the item neighborhood, we can utilize those values in an initial comparison where the predictions by *IdemSvd-Dsvd* will be contrasted against those generated by plain Item-based Filtering. For the latter method, optimal parameter settings were also considered.

From figure 7, which includes the Mean Absolute Errors for *IdemSvd-Dsvd* and plain Item-based Filtering as observed for each of the 5 data splits, we can conclude that demographically enhanced Item-based Filtering, having its demographic vectors reduced by SVD, does indeed provide a considerable accuracy improvement

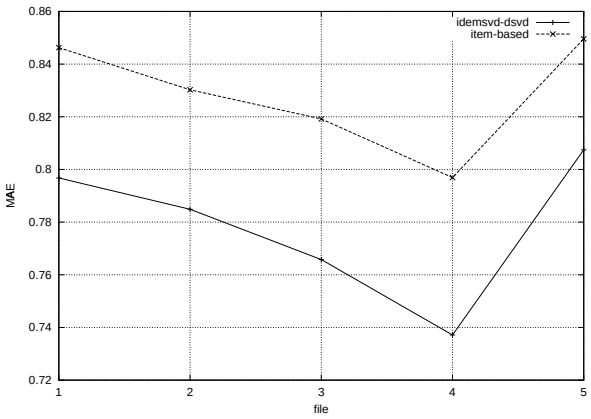

Figure 7. Comparing IdemSvd-Dsvd with plain Item-based Filtering

over plain Item-based Filtering. It is in our intentions to find out which part of this improvement is attributed merely to the participation of the demographic vectors, and which part is owed to the dimensionality reduction applied on them.

7.2.3 Experiments with IdemSvd-Rsvd: Applying SVD only on the user-item matrix

IdemSvd-Rsvd attempts to improve on I-Demog by applying SVD on the user-item matrix. The following experiments intend to, first, identify the optimal settings regarding the size of the neighborhood and the value of k, and then to compare our approach, optimally set, with the best predictions of plain item-based filtering.

Identifying the optimal neighborhood size for IdemSvd-Rsvd

The purpose of this experiment was to identify the best item neighborhood size before contrasting it with plain Item-based Filtering. To achieve that, we had to keep the value of k, corresponding to the low rank user-item matrix, fixed to 6, while setting the enhanced correlation flags according to the values reached in Section 7.2.1. At the same time, we varied the size of the item neighborhood, *neigh-size*=$\{$*10-140*$\}$. Figure 8 displays the generated Mean Absolute Errors (MAEs), averaged over all 5 data splits.

The MAE line in Figure 8 does not follow the behavior commonly detected in similar collaborative filtering experiments, and illustrated in Figure 5, according to which the accuracy improves as the neighborhood size increases, and remains stable or shows a slight decline after surpassing a certain threshold. In the case of IdemSvd-Rsvd, the accuracy starts with a low error value for a neighborhood including 10 items. Then it gets steadily worse for neighborhoods whose size continuously increases, until reaching the maximum size of 140 items. Therefore, the rest of our experiments in this section were executed by defining a neighborhood of 10 items.

Identifying the optimal value of k for IdemSvd-Rsvd

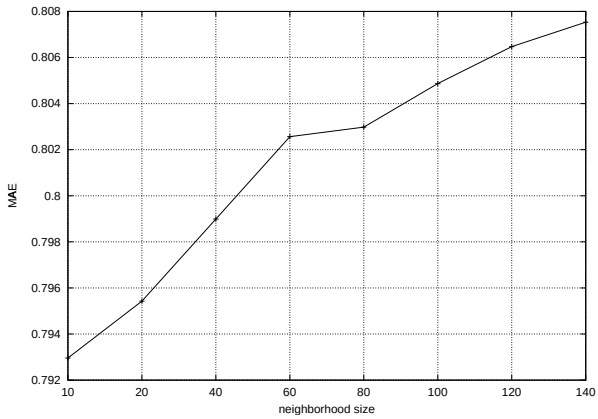

Figure 8. IdemSvd-Rsvd: Identifying the best neighborhood size

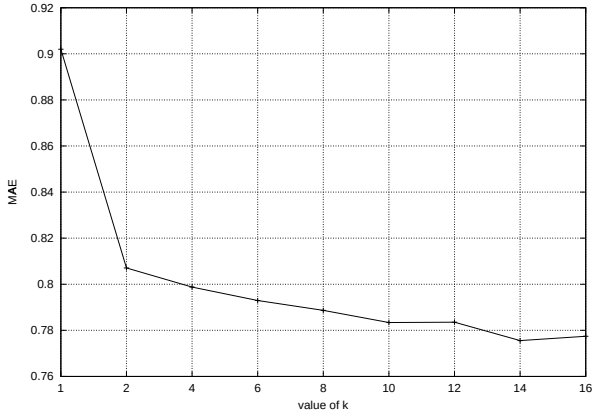

Figure 9. IdemSvd-Rsvd: Identifying the best value of l

Our second experimental step involved trying different values of k, aiming to identify the one that would lead to the best accuracy. The GroupLens data set includes 943 distinct users, and k corresponds to the *pseudo*-users retained after applying SVD in order to reduce the dimensions of the original user-item matrix.

To initiate this experiment we set the neighborhood to its optimal size, according to the previous experiment, and assigned the appropriate values to the enhanced correlation flags. We only varied the values of k, k=$\{$1,2,4,6,8,10,12,14,16$\}$. The generated Mean Absolute Errors (MAEs), averaged over the 5 data splits, are displayed in Figure 9.

For a better understanding of the line in Figure 9 we can view it in correlation with the same experiment in IdemSvd-Dsvd (Figure 6). The demographic matrix, D_i, includes only 18 features, and as a result, in IdemSvd-Dsvd we were able to to reach an optimal accuracy value by retaining only 2 pseudo-features. In the case of IdemSvd-Rsvd, the matrix we are trying to reduce includes a considerably

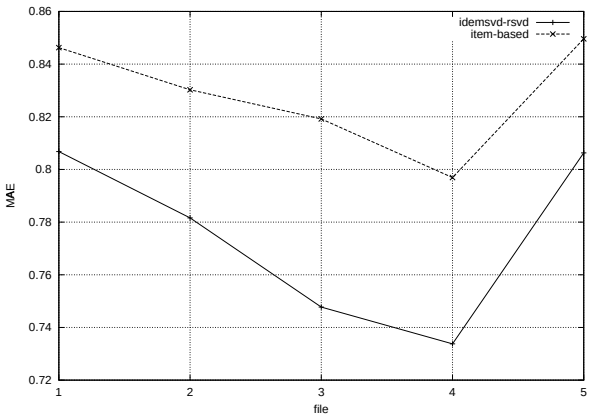

Figure 10. Comparing IdemSvd-Rsvd with plain Item-based Filtering

larger number of users (943>>18). Thus, it is natural to require a comparatively bigger number of pseudo-users in order to capture a behavior which approximates optimally the behavior of the original matrix. Specifically, as shown from the figure, a value of k equal to 14 provides the best system accuracy.

Comparing IdemSvd-Rsvd with plain Item-based Filtering

We have now identified the optimal parameter settings for k and the size of the item neighborhood. We can apply those values on *IdemSvd-Rsvd* and compare the generated predictions with those produced by plain Item-based Filtering, when also optimally tuned.

The obvious conclusion from Figure 10, which includes the Mean Absolute Errors for *IdemSvd-Rsvd* and plain Item-based Filtering as observed for each of the 5 data splits, is that demographically enhanced Item-based Filtering, with its original user-item matrix reduced by SVD, does indeed provide a considerable accuracy improvement over plain Item-based Filtering. With further experiments we intend to specify how these results fare against other approaches which feature demographically enhanced Item-based Filtering, but allow SVD to affect different aspects of their filtering process.

7.2.4 Experiments with IdemSvd-2svd: Applying SVD on both demographic and user-item matrix

IdemSvd-2svd can be characterized as demographically enhanced Item-based Filtering, which has both its demographic and user-item matrix reduced by SVD. The corresponding experiments include an additional step, when contrasted to those on IdemSvd-Dsvd and IdemSvd-Rsvd, since the parameters we have to tune before we compare this approach against plain item-based filtering, include (i) the size of the neighborhood, (ii) the value of k, which represents the dimensions retained after applying SVD on the user-item matrix, and (iii) the value of l, which represents the dimensions retained after applying SVD on the demographic vectors.

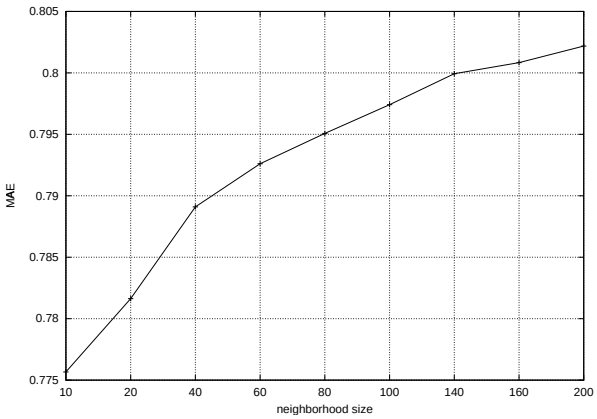

Figure 11. IdemSvd-2svd: Identifying the best neighborhood size

Identifying the optimal neighborhood size for IdemSvd-2svd

In the past two sections we tested the effect of a neighborhood size change for two implementations of IdemSvd (*IdemSvd-Dsvd* and *IdemSvd-Rsvd*), which are similar in that they apply SVD on a single point of the filtering procedure, but differ in that point of application. The experimental results were completely opposite. We could now run the same experiment on a filtering approach which applies SVD on two points of the procedure and observe its behavior. To achieve that, we set the values of k and l to 14 and 2 respectively, and only varied the size of the item neighborhood (*neigh-size=*{*10-200*}). Figure 11 displays the generated Mean Absolute Errors, averaged over the 5 data splits.

The behavior of IdemSvd-2svd, as defined by the line in Figure 11, differs from what reported for plain Item-based Filtering, and also by the corresponding experiment on IdemSvd-Dsvd (Figure 5). The changes in the size of the neighborhood affect the system's accuracy in a way directly comparable to IdemSvd-Rsvd, which applies SVD only on the user-item matrix. Specifically, the lowest error values were observed for the smallest neighborhood size tested in our experiments (*neigh-size=10*). As the size of the neighborhood was getting bigger, until reaching its maximum size (*neigh-size=200*), the recorded error kept increasing. Therefore, the neighborhood selected for the subsequent experiments includes 10 items.

Identifying the optimal value of k for IdemSvd-2svd

Our second experiment with IdemSvd-2svd involved testing the algorithm for different values of k, which expressed distinct low rank representations of the original user-item matrix. According to the results of the previous experiment, we defined an optimal neighborhood of 10 items. We also assigned to l a random value of 2. We only varied the values of k, k={1,2,4,6,8,10,12,14,16,18}. The generated Mean Absolute Errors (MAEs), averaged over the 5 data splits, are displayed in Figure 12.

In a result far from surprising, and similarly to the observations included in the previous experiment, the behavior of IdemSvd-2svd, when varying the value of k,

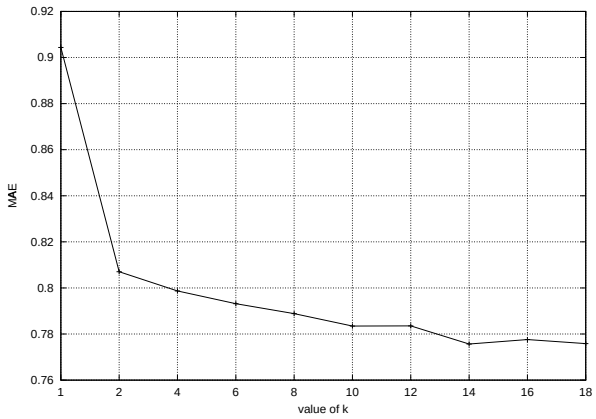

Figure 12. IdemSvd-2svd: Identifying the best value of k

appears to be very close to the one reported by the same experiment in IdemSvd-Rsvd (Figure 9). In both cases we are defining the number of *pseudo*-users to retain, out of the 943 existing in the initial user-item matrix. Furthermore, the behavior of the line, which depicts a decrease of MAE as the value of k increases, makes sense: it only seems natural that a bigger k would allow for a better approximation of the original user-item space. According to these experimental results any value in the range of k=$\{$14-18$\}$ can be assigned to k without considerable, if any, accuracy loss.

Identifying the optimal value of l for IdemSvd-2svd

With our third IdemSvd-2svd experiment, we wanted to identify the optimal number of *pdeudo*-features to retain from the original, 18-featured demographic vectors, expressed by l. We utilized the optimal values for k and the size of the neighborhood, as obtained from the past 2 experiments, and only varied the values of l, l=$\{$1,2,4,6,8,10,12,14,16$\}$. Figure 13 depicts the generated Mean Absolute Errors, averaged over all 5 data splits.

We can easily conclude that there is no specific pattern followed by the MAE line: the error moves up and down repeatedly, reaching its lowest value for l=8. This behavior is different from the one in Figure 6, which depicted the error of the same experiment in IdemSvd-Dsvd. We should also note that the effect of l on the system accuracy is rather trivial. Any performance variation caused by it, is limited to the third decimal place of the MAE values, allowing us to describe it as a fine-tuning of accuracy results which were achieved by the previous experiments.

Comparing IdemSvd-2svd with plain Item-based Filtering

At this point we have identified the optimal parameter settings for k, l, and the size of the item neighborhood. We will apply those values on *IdemSvd-2svd* in an attempt to compare its optimal predictions against those generated by plain Item-based Filtering. For the latter method, optimal parameter settings were also utilized.

Figure 14 records the Mean Absolute Errors of *IdemSvd-2svd* and plain Item-

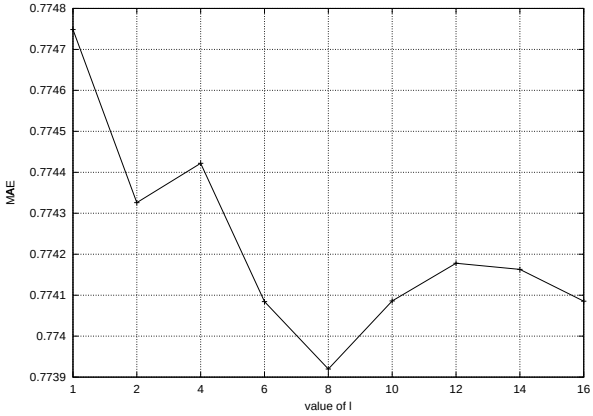

Figure 13. IdemSvd-2svd: Identifying the best value of l

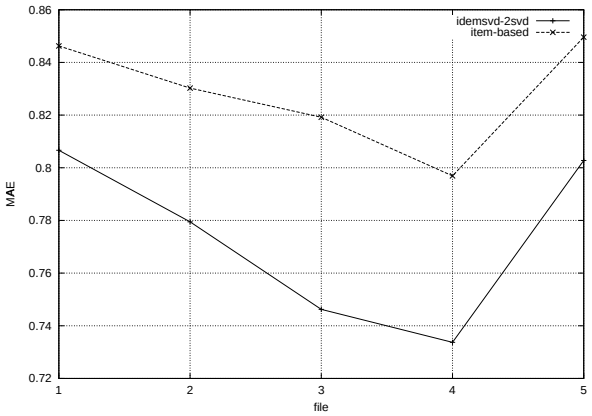

Figure 14. Comparing IdemSvd-2svd with plain Item-based Filtering

based Filtering for each of the 5 data splits. Based on these MAE lines we can claim that demographically enhanced Item-based Filtering, with SVD applied to both user-item and demographic matrices, does indeed provide a considerable accuracy improvement over plain Item-based Filtering.

7.2.5 Overall comparison of IdemSvd implementations

The aim of this final section was to collect the most accurate results generated by the 4 *IdemSvd* implementations (*I-Demog*, *IdemSvd-Dsvd*, *IdemSvd-Rsvd* and *IdemSvd-2svd*) and utilize them in an overall comparison. According to the methodology followed in preceding sections, we selected plain Item-based CF as our base algorithm.

Figure 15 depicts the lowest Mean Absolute Errors which were reported in each of these cases, for the 5 splits of the data set. The averages of these MAEs, which

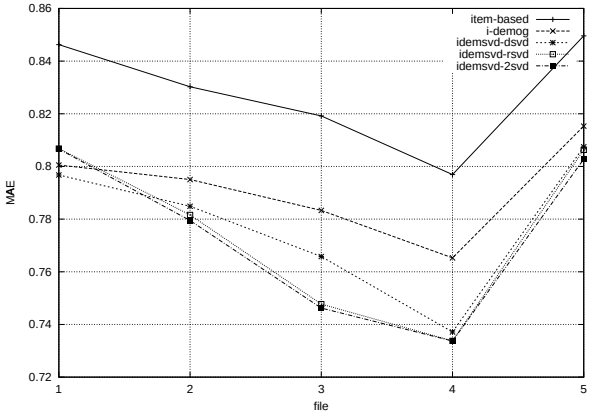

Figure 15. Comparing all IdemSvd implementations

Table 4. Best IdemSvd MAEs averaged over 5 data splits

	item-based	**i-demog**	**idemsvd-dsvd**	**idemsvd-rsvd**	**idemsvd-2svd**
MAE	0.82843656	0.79189590	0.77840260	0.77523445	0.77376098

represent a single best mean accuracy value for each participating approach, are included in Table 4. Table 5 collects the corresponding execution costs. We note that m is the number of users, n is the number of items, and k is the number of *pseudo*-users retained after the application of SVD, where $k << n$.

In the cases of *item-based* and *i-demog*, there is no application of SVD. At the same time, and contrary to what happens in the user-based approaches, the item-item correlation matrix is a lot less volatile. This means that we can compute the item correlations in less frequently intervals without affecting the overall performance of the system, which allows us to assign them to the off-line component. This move places an mn^2 cost under the *off-line* column. The *on-line* component can be now dedicated solely to prediction generation, which in the worse case induces an n^2 cost. Still, this cost is, on average, reduced to nl, where l corresponds to the size of the item neighborhood, which is usually $l << n$.

Regarding the 3 remaining methods (*idemsvd-dsvd*, *idemsvd-rsvd* and *idemsvd-2svd*), they all involve the application of SVD. The additional cost is equal to

Table 5. Off and On-line costs for all 5 methods

	Off-line	**On-line**
item-based	mn^2	n^2
i-demog	mn^2	n^2
idemsvd-dsvd	$[m^3]+[n^2m]$	n^2
idemsvd-rsvd	$[m^2n + m^3]+[n^2k]$	n^2
idemsvd-2svd	$[2(m^2n + m^3)]+[n^2k]$	n^2

$m^2n + m^3$ for IdemSvd-Rsvd and equal to $2(m^2n + m^3)$ for IdemSvd-2svd, since SVD is applied twice. The situation is a bit more complicated for IdemSvd-Dsvd: we apply SVD on an $m \times 18$ matrix, of m items over 18 item demographic features. This leads to a cost of $m^2n + m^3 = 18m^2 + m^3 = m^3$.

For the cases of IdemSvd-Rsvd and IdemSvd-2svd, there are only k *pseudo*-users involved in the item similarities calculations after the application of SVD, which means that the corresponding costs would be reduced to n^2k. Still, and contrary to what discussed in user-based approaches, this reduction stays at the off-line component, and cannot be fully realized. The same cost remains at n^2m for IdemSvd-Dsvd, where no SVD is applied on the user-item matrix.

In the case of the GroupLens data set, which we utilized for our experiments, m=943 and n=1682. Therefore, we can assume that m and n are of the same order and adjust the execution costs of *IdemSvd-Rsvd* and *IdemSvd-Dsvd* to approximately n^3, and of *IdemSvd-2svd* to $2n^3$. The on-line costs could be adapted accordingly.

Bearing in mind the accuracy results from Figure 15 and Table 4, along with the execution costs from Table 5, we can reach the following conclusions:

- *Plain Item-based Collaborative Filtering* has the lowest off-line execution cost, along with I-Demog. Still, it cannot be recommended since its accuracy ranks at the last place among those tested.

- *I-Demog*'s off-line cost is equal to that of plain Item-based CF, but its accuracy is significantly improved. Thus, we should prefer it in cases where we cannot handle the burden placed on the off-line component by the execution of SVD.

- Comparing the 2 filtering approaches which apply SVD *once* during the filtering procedure, we should select *IdemSvd-Dsvd* if we care for the lowest off-line cost, and *IdemSvd-Rsvd* if we prefer the best accuracy.

- *IdemSvd-2svd* may be the method with the lowest overall error, but, at the same time, it incurs the biggest off-line costs. Thus, among all approaches which apply SVD, we should prefer IdemSvd-2svd only when those off-line costs won't matter when compared to the improvement in the system's performance, or when we are able to calculate the off-line component at the least frequent time intervals.

- Conclusively, and contrary to the results reported after executing the same experiments in User-based CF, we can claim that SVD can successfully enhance the I-Demog algorithm. All 3 approaches we tested lead to a performance improvement over plain Item-based CF or I-Demog. Thus, we can recommend any of them under different circumstances, depending on the priorities we have set regarding the frequency of execution of the off-line component, the time which can be designated for the off-line component, and the prediction accuracy which can be achieved.

8 Conclusions and Future Work

In the past, Singular Value Decomposition was mainly combined with User-based Collaborative Filtering, proving to be an effective solution for recorded problems of Recommender Systems such as scalability and sparsity. By this work we extended the application of SVD to Item-based Collaborative Filtering. 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. We also tested its effectiveness when combined with data other than the common user ratings on items, by utilizing it in collaboration with demographic information. Our proposed methods can be described as filtering algorithms which utilize SVD in order to reduce the dimensions of the original user-item and/or demographic matrix, while, at the same time drawing supplementary information possibly available in related demographic data.

A number of experiments were set up to check how our proposed algorithms fare against Item-based filtering, running in collaboration with the original data representation. The results in both cases showed that low-dimension Item-based Filtering not only alleviates problems like scalability or sparsity of the data, but also proves to be a more accurate recommender than the original Item-based Filtering. Furthermore, we noted that the inclusion of related demographic data proved capable of leading to a recommender system which further improves its effectiveness, when compared to Item-based Filtering merely enhanced by SVD.

Keeping these promising results as our starting point, it is in our intentions to experiment with Principal Component Analysis, as a viable alternative to SVD. We view it as a second method which can possibly enhance the filtering procedure, by assisting in dimensionality reduction. Furthermore, Artificial Neural Networks employing pseudo-SVD as a possibly simpler, faster and less costly solution, would be an interesting alternative that we can compare our algorithm against.

References

[1] P. Melville, R. J. Mooney, and R. Nagarajan, "Content-boosted collaborative filtering," in *ACM SIGIR Workshop on Recommender Systems*, New Orleans, LA, 2001.

[2] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin, "Combining content-based and collaborative filters in an online newspaper," in *ACM SIGIR Workshop on Recommender Systems-Implementation and Evaluation*, Berkeley, CA, 1999.

[3] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. T. Riedl, "Analysis of recommendation algorithms for e-commerce," in *Electronic Commerce*, 2000.

[4] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society for Information Science*, vol. 41, no. 6, pp. 391–407, 1990.

[5] M. W. Berry, S. T. Dumais, and G. W. O'Brien, "Using linear algebra for intelligent information retrieval," *SIAM Review*, vol. 37, pp. 573–595, 1995.

[6] D. Billsus and M. J. Pazzani, "Learning collaborative information filters," in *15th International Conference on Machine Learning*, Madison, WI, 1998.

[7] B. M. Sarwar, "Sparsity, scalability, and distribution in recommender systems," Ph.D. dissertation, University of Minnesota, 2001.

[8] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Incremental singular value decomposition algorithms for highly scalable recommender systems," in *Fifth International Conference on Computer and Information Technology (ICCIT 2002)*, 2002.

[9] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, "Eigentaste: A constant time collaborative filtering algorithm," *Information Retrieval Journal*, vol. 4, pp. 133–151, 2001.

[10] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. T. Riedl, "Item-based collaborative filtering recommendation algorithms," in *10th International World Wide Web Conference (WWW10)*, Hong Kong, 2001.

[11] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Transactions on Information Systems*, vol. 22, pp. 5–53, 2004.

[12] E. G. Vozalis and K. G. Margaritis, "Recommender systems: An experimental comparison of two filtering algorithms," in *Proceedings of the 9th Panhellenic Conference in Informatics - PCI 2003*, 2003.

[13] M. Vozalis and K. G. Margaritis, "Collaborative filtering enhanced by demographic correlation," in *Proceedings of the AIAI Symposium on Professional Practice in AI, part of the 18th World Computer Congress*, Toulouse, France, 2004.