Content-based Dimensionality Reduction for Recommender Systems

Panagiotis Symeonidis

Aristotle University, Department of Informatics, Thessaloniki 54124, Greece
symeon@csd.auth.gr

Abstract. Recommender Systems are gaining widespread acceptance in e-commerce applications to confront the information overload problem. Collaborative Filtering (CF) is a successful recommendation technique, which is based on past ratings of users with similar preferences. In contrast, Content-based Filtering (CB) exploits information solely derived from document or item features (e.g. terms or attributes). CF has been combined with CB to improve the accuracy of recommendations. A major drawback in most of these hybrid approaches was that these two techniques were executed independently. In this paper, we construct a feature profile of a user based on both collaborative and content features. We apply Latent Semantic Indexing (LSI) to reveal the dominant features of a user. We provide recommendations according to this dimensionally-reduced feature profile. We perform experimental comparison of the proposed method against well-known CF, CB and hybrid algorithms. Our results show significant improvements in terms of providing accurate recommendations.

1 Introduction

Collaborative Filtering (CF) is a successful recommendation technique. It is based on past ratings of users with similar preferences, to provide recommendations. However, this technique introduces certain shortcomings. For instance, if a new item appears in the database, there is no way to be recommended before it is rated.

In contrast, Content-Based filtering (CB) exploits only information derived from document or item features (e.g., terms or attributes). Latent Semantic Indexing (LSI) has been extensively used in the CB field, in detecting the latent semantic relationships between terms and documents. LSI constructs a low-rank approximation to the term-document matrix. As a result, it produces a less noisy matrix which is better than the original one. Thus, higher level concepts are generated from plain terms.

Recently, CB and CF have been combined to improve the recommendation procedure. Most of these hybrid systems are process-oriented: they run CF on the results of CB and vice versa. CF exploits information from the users and

their ratings. CB exploits information from items and their features. However being hybrid systems, they miss the interaction between user ratings and item features.

In this paper, we construct a feature profile of a user to reveal the duality between users and features. For instance, in a movie recommender system, a user prefers a movie for various reasons, such as the actors, the director or the genre of the movie. All these features affect differently the choice of each user. Then, we apply Latent Semantic Indexing Model (LSI) to reveal the dominant features of a user. Finally, we provide recommendations according to this dimensionally-reduced feature profile. Our experiments with a real-life data set show the superiority of our approach over existing CF, CB and hybrid approaches.

The rest of this paper is organized as follows: Section 2 summarizes the related work. The proposed approach is described in Section 3. Experimental results are given in Section 4. Finally, Section 5 concludes this paper.

2 Related Work

In 1994, the GroupLens system implemented a CF algorithm based on common users preferences. Nowadays, this algorithm is known as user-based CF. In 2001, another CF algorithm was proposed. It is based on the items' similarities for a neighborhood generation. This algorithm is denoted as item-based CF.

The Content-Based filtering approach has been studied extensively in the Information Retrieval (IR) community. Recently, Schult and Spiliopoulou (2006) proposed the Theme-Monitor algorithm for finding emerging and persistent themes in document collections. Moreover, in IR area, Furnas et al. (1988) proposed LSI to detect the latent semantic relationship between terms and documents. Sarwar et al. (2000) applied dimensionality reduction for the user-based CF approach.

There have been several attempts to combine CB with CF. The Fab System (Balabanovic et al. 1997), measures similarity between users after first computing a content profile for each user. This process reverses the CinemaScreen System (Salter et al. 2006) which runs CB on the results of CF. Melville et al. (2002) used a content-based predictor to enhance existing user data, and then to provide personalized suggestions though collaborative filtering. Finally, Tso and Schmidt-Thieme (2005) proposed three attribute-aware CF methods applying CB and CF paradigms in two separate processes before combining them at the point of prediction.

All the aforementioned approaches are hybrid: they either run CF on the results of CB or vice versa. Our model, discloses the duality between user ratings and item features, to reveal the actual reasons of their rating behavior. Moreover, we apply LSI on the feature profile of users to reveal the principal features. Then, we use a similarity measure which is based on features, revealing the real preferences of the user's rating behavior.

3 The Proposed Approach

Our approach constructs a feature profile of a user, based on both collaborative and content features. Then, we apply LSI to reveal the dominant features trends. Finally, we provide recommendations according to this dimensionally-reduced feature profile of the users.

3.1 Defining Rating, Item and Feature Profiles

CF algorithms process the rating data of the users to provide accurate recommendations. An example of rating data is given in Figures 1a and 1b. As shown, the example data set (Matrix R) is divided into a training and test set, where I_{1-12} are items and U_{1-4} are users. The null cells (no rating) are presented with dash and the rating scale is between [1-5] where 1 means strong dislike, while 5 means strong like.

Definition 1 *The rating profile $R(U_k)$ of user U_k is the k-th row of matrix R.*

For instance, $R(U_1)$ is the rating profile of user U_1, and consists of the rated items I_1,I_2,I_3,I_4,I_8 and I_{10}. The rating of a user u over an item i is given from the element $R(u, i)$ of matrix R.

	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}	I_{12}
U_1	5	3	5	4	-	1	-	3	-	5	-	-
U_2	3	-	-	-	4	5	1	-	5	-	-	1
U_3	1	-	5	4	5	-	5	-	-	3	5	-

(a)

	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}	I_{12}
U_4	5	-	1	-	-	4	-	-	3	-	-	5

(b)

	f_1	f_2	f_3	f_4
I_1	1	1	0	0
I_2	1	0	0	0
I_3	1	0	1	1
I_4	1	0	0	1
I_5	0	1	1	0
I_6	0	1	0	0
I_7	0	0	1	1
I_8	0	0	0	1
I_9	0	1	1	0
I_{10}	0	0	0	1
I_{11}	0	0	1	1
I_{12}	0	1	0	0

(c)

Fig. 1. (a) Training Set ($n \times m$) of Matrix R, (b) Test Set of Matrix R, (c) Item-Feature Matrix F

As described, content data are provided in the form of features. In our running example illustrated in Figure 1c for each item we have four features that describe its characteristics. We use matrix F, where element $F(i,\ f)$ is one, if item i contains feature f and zero otherwise.

Definition 2 *The item profile $F(I_k)$ of item I_k is the k-th row of matrix F.*

For instance, $F(I_1)$ is the profile of item I_1, and consists of features F_1 and F_2. Notice that this matrix is not always boolean. Thus, if we process documents, matrix F would count frequencies of terms.

To capture the interaction between users and their favorite features, we construct a feature profile composed of the rating profile and the item profile.

For the construction of the feature profile of a user, we use a positive rating threshold, P_τ, to select items from his rating profile, whose rating is not less than this value. The reason is that the rating profile of a user consists of ratings that take values from a scale(in our running example, 1-5 scale). It is evident that ratings should be "positive", as the user does not favor an item that is rated with 1 in a 1-5 scale.

Definition 3 *The feature profile $P(U_k)$ of user U_k is the k-th row of matrix P whose elements $P(u,f)$ are given by Equation 1.*

$$P(u,f) = \sum_{\forall R(u,i) > P_\tau} F(i,f) \tag{1}$$

In Figure 2, element $P(U_k,f)$ denotes an association measure between user U_k and feature f. In our running example (with $P_\tau = 2$), P(U_2) is the feature profile of user U_2, and consists of features f_1, f_2 and f_3. The correlation of a user U_k over a feature f is given from the element $P(U_k, f)$ of matrix P. As shown, feature f_2 describe him better, than feature f_1 does.

	f_1	f_2	f_3	f_4
U_1	4	1	1	4
U_2	1	4	2	0
U_3	2	1	4	5

(a)

	f_1	f_2	f_3	f_4
U_4	1	4	1	0

(b)

Fig. 2. User-Feature matrix P divided in (a) Training Set ($n \times m$), (b) Test Set

3.2 Applying SVD on Training Data

Initially, we apply Singular Value Decomposition (SVD) on the training data of matrix P that produces three matrices based on Equation 2, as shown in Figure 3:

$$P_{n\times m} = U_{n\times n} \cdot S_{n\times m} \cdot V'_{m\times m} \tag{2}$$

4	1	1	4
1	4	2	0
2	1	4	5

$P_{n\times m}$

-0.61	0.28	-0.74
-0.29	-0.95	-0.12
-0.74	0.14	0.66

$U_{n\times n}$

8.87	0	0	0
0	4.01	0	0
0	0	2.51	0

$S_{n\times m}$

-0.47	-0.28	-0.47	-0.69
0.11	-0.85	-0.27	0.45
-0.71	-0.23	0.66	0.13
-0.52	0.39	-0.53	0.55

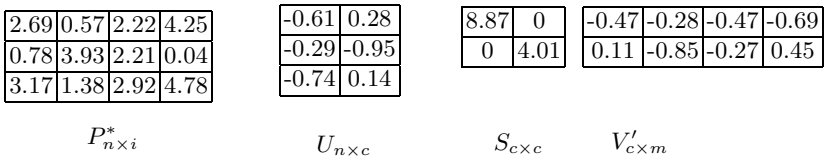
$V'_{m\times m}$

Fig. 3. Example of: $P_{n\times m}$ (initial matrix P), $U_{n\times m}$ (left singular vectors of P), $S_{n\times m}$ (singular values of P), $V'_{m\times m}$ (right singular vectors of P).

3.3 Preserving the Principal Components

It is possible to reduce the $n \times m$ matrix S to have only c largest singular values. Then, the reconstructed matrix is the closest rank-c approximation of the initial matrix P as it is shown in Equation 3 and Figure 4:

$$P^*_{n\times m} = U_{n\times c} \cdot S_{c\times c} \cdot V'_{c\times m} \tag{3}$$



$P^*_{n\times i}$ $U_{n\times c}$ $S_{c\times c}$ $V'_{c\times m}$

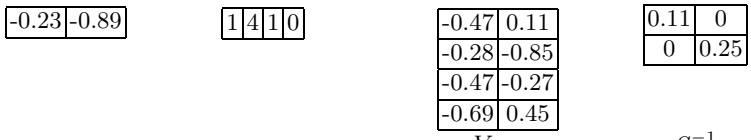
Fig. 4. Example of: $P^*_{n\times m}$ (approximation matrix of P), $U_{n\times c}$ (left singular vectors of P^*), $S_{c\times c}$ (singular values of P^*), $V'_{c\times m}$ (right singular vectors of P^*).

We tune the number, c, of principal components (i.e., dimensions) with the objective to reveal the major feature trends. The tuning of c is determined by the information percentage that is preserved compared to the original matrix.

3.4 Inserting a Test User in the c-dimensional Space

Given the current feature profile of the test user u as illustrated in Figure 2b, we enter pseudo-user vector in the c-dimensional space using Equation 4. In our example, we insert U_4 into the 2-dimensional space, as shown in Figure 5:

$$\mathbf{u_{new}} = \mathbf{u} \cdot V_{m\times c} \cdot S^{-1}_{c\times c} \tag{4}$$



$\mathbf{u_{new}}$ $\mathbf{u}$ $V_{m\times c}$ $S^{-1}_{c\times c}$

Fig. 5. Example of: $\mathbf{u_{new}}$ (inserted new user vector), $\mathbf{u}$ (user vector), $V_{m\times c}$ (two left singular vectors of V), $S^{-1}_{c\times c}$ (two singular values of inverse S).

In Equation 4, $\mathbf{u_{new}}$ denotes the mapped ratings of the test user $\mathbf{u}$, whereas $V_{m\times c}$ and $S^{-1}_{c\times c}$ are matrices derived from SVD. This $\mathbf{u_{new}}$ vector should be added in the end of the $U_{n\times c}$ matrix which is shown in Figure 4.

3.5 Generating the Neighborhood of users/items

In our model, we find the k nearest neighbors of pseudo user vector in the c-dimensional space. The similarities between train and test users can be based on Cosine Similarity. First, we compute the matrix $U_{n\times c} \cdot S_{c\times c}$ and then we perform vector similarity. This $n \times c$ matrix is the c-dimensional representation for the n users.

3.6 Generating the top-N recommendation list

The most often used technique for the generation of the top-N list, is the one that counts the frequency of each positively rated item inside the found neighborhood, and recommends the N most frequent ones. Our approach differentiates from this technique by exploiting the item features. In particular, for each feature f inside the found neighborhood, we add its frequency. Then, based on the features that an item consists of, we count its weight in the neighborhood. Our method, takes into account the fact that, each user has his own reasons for rating an item.

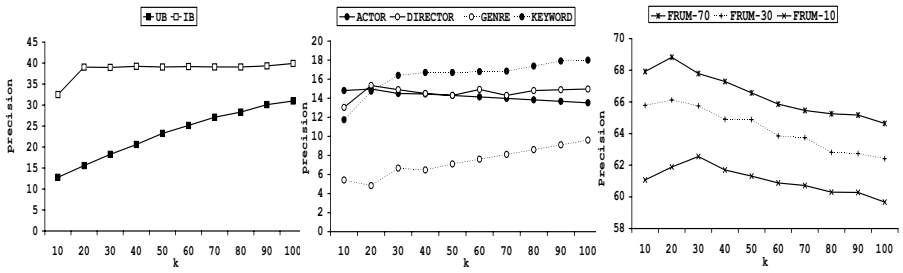
4 Performance Study

In this section, we study the performance of our Feature-Weighted User Model (FRUM) against the well-known CF, CB and a hybrid algorithm. For the experiments, the collaborative filtering algorithm is denoted as CF and the content-based algorithm as CB. As representative of the hybrid algorithms, we used the Cinemascreen Recommender Agent (SALTER et al. 2006), denoted as CFCB. Factors that are treated as parameters, are the following: the neighborhood size (k, default value 10), the size of the recommendation list (N, default value 20) and the size of train set (default value 75%). P_τ threshold is set to 3. Moreover, we consider the division between training and test data. Thus, for each transaction of a test user we keep the 75% as hidden data (the data we want to predict) and use the rest 25% as not hidden data (the data for modeling new users). The extraction of the content features has been done through the well-known internet movie database (imdb). We downloaded the plain imdb database (ftp.fu-berlin.de - October 2006) and selected 4 different classes of features (genres, actors, directors, keywords). Then, we join the imdb and the Movielens data sets. The joining process lead to 23 different genres, 9847 keywords, 1050 directors and 2640 different actors and actresses (we selected only the 3 best paid actors or actresses for each movie). Our evaluation metrics are from the information retrieval field. For a test user that receives a top-N recommendation list, let R denote the number of *relevant recommended items* (the items of the top-N list that are rated higher than P_τ by the test user). We define the following: *Precision* is the ratio of R to N.*Recall* is the ratio of R to the total number of relevant items for the test user (all items rated higher than P_τ by him). In the following, we also use $\mathrm{F}_1 = 2 \cdot \mathrm{recall} \cdot \mathrm{precision}/(\mathrm{recall} + \mathrm{precision})$. F_1 is used because it combines both precision and recall.

4.1 Comparative Results for CF, CB, CFCB and FRUM Algorithms

For the CF algorithms, we compare the two main cases, denoted as user-based (UB) and item-based (IB) algorithms. The former constructs a user-user

similarity matrix while the latter, builds an item-item similarity matrix. Both of them, exploit the user ratings information(user-item matrix R). Figure 6a demonstrates that IB compares favorably against UB for small values of k. For large values of k, both algorithms converge, but never exceed the limit of 40% in terms of precision. The reason is that as the k values increase, both algorithms tend to recommend the most popular items. In the sequel, we will use the IB algorithm as a representative of CF algorithms.



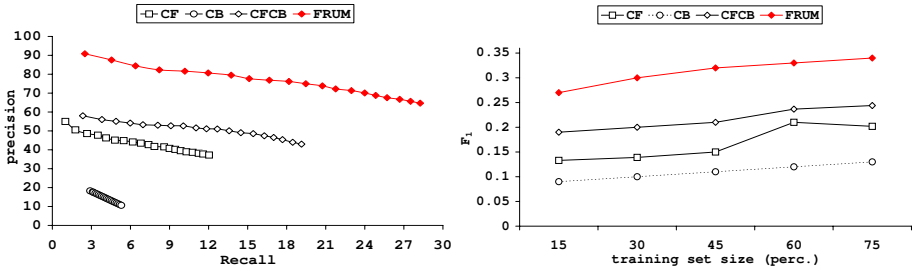
(a) (b) (c)

Fig. 6. Precision vs. k of: (a) UB and IB algorithms, (b) 4 different feature classes, (c) 3 different information percentages of our FRUM model

For the CB algorithms, we have extracted 4 different classes of features from the imdb database. We test them using the pure content-based CB algorithm to reveal the most effective in terms of accuracy. We create an item-item similarity matrix based on cosine similarity applied solely on features of items (item-feature matrix F). In Figure 6b, we see results in terms of precision for the four different classes of extracted features. As it is shown, the best performance is attained for the "keyword" class of content features, which will be the default feature class in the sequel.

Regarding the performance of our FRUM, we preserve, each time, a different fraction of principal components of our model. More specifically, we preserve 70%, 30% and 10% of the total information of initial user-feature matrix P. The results for precision vs. k are displayed in Figure 6c. As shown, the best performance is attained with 70% of the information preserved. This percentage will be the default value for FRUM in the sequel.

In the following, we test FRUM algorithm against CF, CB and CFCB algorithms in terms of precision and recall based on their best options. In Figure 7a, we plot a precision versus recall curve for all four algorithms. As shown, all algorithms' precision falls as N increases. In contrast, as N increases, recall for all four algorithms increases too. FRUM attains almost 70% precision and 30% recall, when we recommend a top-20 list of items. In contrast, CFCB attains 42% precision and 20% recall. FRUM is more robust in finding relevant items to a user. The reason is two-fold:(i) the sparsity has been downsized through the features and (ii) the LSI application reveals the dominant feature trends.



(a) (b)

Fig. 7. Comparison of CF, CB, CFCB with FRUM in terms of (a) precision vs. recall (b) training set size.

Now we test the impact of the size of the training set. The results for the F_1 metric are given in Figure 7b. As expected, when the training set is small, performance downgrades for all algorithms. FRUM algorithm is better than the CF, CB and CFCB in all cases. Moreover, low training set sizes do not have a negative impact on measure F_1 of the FRUM algorithm.

5 Conclusions

We propose a feature-reduced user model for recommender systems. Our approach builds a feature profile for the users, that reveals the real reasons of their rating behavior. Based on LSI, we include the pseudo-feature user concept in order to reveal his real preferences. Our approach outperforms significantly existing CF, CB and hybrid algorithms. In our future work, we will consider the incremental update of our model.

References

BALABANOVIC, M. and SHOHAM, Y. (1997): Fab: Content-based, collaborative recommendation, *ACM Communications*,volume 40,number 3,66-72

FURNAS, G. and DEERWESTER, et al. (1988): Information retrieval using a singular value decomposition model of latent semantic structure, *SIGIR* , 465-480

MELVILLE, P. and MOONEY R. J. and NAGARAJAN R. (2002): Content-Boosted Collaborative Filtering for Improved Recommendations, *AAAI*, 187-192

SALTER, J. and ANTONOPOULOS, N. (2006): CinemaScreen Recommender Agent: Combining Collaborative and Content-Based Filtering *Intelligent Systems Magazine*, volume 21, number 1, 35-41

SARWAR, B. and KARYPIS, G. and KONSTAN, J. and RIEDL, J. (2000) Application of dimensionality reduction in recommender system-A case study", *ACM WebKDD Workshop*

SCHULT, R and SPILIOPOULOU, M. (2006) : Discovering Emerging Topics in Unlabelled Text Collections *ADBIS 2006*, 353-366

TSO, K. and SCHMIDT-THIEME, L. (2005) : Attribute-aware Collaborative Filtering, *German Classification Society GfKl 2005*