Feature-Weighted User Model for Recommender Systems[⋆]

Panagiotis Symeonidis, Alexandros Nanopoulos, and Yannis Manolopoulos

Aristotle University, Department of Informatics, Thessaloniki 54124, Greece
{symeon, alex, manolopo}@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) assumes that each user operates independently. As a result, it exploits only information derived from document or item features. Both approaches have been extensively combined to improve the recommendation procedure. Most of these systems are hybrid: 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. In this paper, we construct a feature-weighted user profile to disclose the duality between users and features. Exploiting the correlation between users and features we reveal the real reasons of their rating behavior. We perform experimental comparison of the proposed method against the well-known CF, CB and a hybrid algorithm with a real data set. Our results show significant improvements, in terms of effectiveness.

1 Introduction

Recommender systems are gaining widespread acceptance in e-commerce and other world wide web applications to confront the "information overload" problem. It is recognized that user modeling plays the main role in the success of these systems [2]. A robust user model should handle several real life problems such as, the sparsity of data, the over-specialization, the shallow analysis of content, the unwillingness of users to fill in their profile and so on.

Collaborative Filtering (CF) and memory-based (nearest-neighbor) algorithms in particular, are successful recommendation techniques. They are based on past ratings of users with similar preferences, to provide recommendations [5]. However, this technique introduces certain shortcomings. If a new item appears in the database, there is no way to be recommended before it is rated. On the other hand, if a user's taste is unusual, he can not find neighbors, and gets inaccurate recommendations.

In contrast, Content-Based filtering (CB) assumes that each user operates independently. As a result, CB exploits only information derived from document or

⋆ This paper is supported by a national GSRT PABET-NE project.

item features (e.g., terms or attributes). A pure content-based system faces the problem of over-specialization [2], where a user is restricted to seeing items similar to those already rated. It also suffers from possible shallow analysis of content.

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-weighted user profile to disclose the duality between users and features. Moreover, exploiting the correlation between users and features we reveal the actual reasons of their rating behavior. 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. Our approach correlates user ratings with item features bringing to surface the actual reasons of user preferences.

Our contribution is summarized as follows: (i) We construct a novel feature-weighted user model, which discloses the duality between users and features, (ii) based on Information Retrieval, we include the Term Frequency Inverse Document Frequency (TFIDF) weighting scheme in CF, (iii) we propose a new top-N generation list algorithm based on features frequency and (iv) we perform extensive experimental results with the internet movies database (imdb) and MoviesLens data sets, which demonstrate the superiority of the proposed approach.

The rest of this paper is organized as follows: Section 2 summarizes the related work, whereas Section 3 contains the analysis of the examined factors. The proposed approach is described in Section 4. Experimental results are given in Section 5. Finally, Section 6 concludes this paper.

2 Related Work

In 1994, the GroupLens system [5] implemented a CF algorithm based on common users preferences. Nowadays, this algorithm is known as user-based CF, because it employs users' similarities for the formation of the neighborhood of nearest users. In 2001, another CF algorithm was proposed. It is based on the items' similarities for neighborhood generation [7]. This algorithm is denoted as item-based or item-item CF, because it employs items' similarities for the formation of the neighborhood of nearest users.

The content-based approach has been studied in the information retrieval (IR) community. It assumes that each user operates independently. It exploits only information derived from documents or item features (eg. terms or attributes). There are two basic subproblems in designing a content filtering system: (i) It is the user profile construction and (ii) the document representation. In 1994, Yan et al. [8] implemented a simple content-based filtering system for internet news articles (SIFT).

There have been several attempts to combine CB with CF. The Fab System [2], measures similarity between users after first computing a profile for each user. This process reverses the CinemaScreen System [6] which runs CB on the results of CF. Melville et al. [4] used a content-based predictor to enhance existing user data, and then to provide personalized suggestions though collaborative filtering. Finally, Xin Jin et al. [3] proposed a Web recommendation system in which collaborative and content features are integrated under the maximum entropy principle.

All the aforementioned approaches are hybrid: they either run CF on the results of CB or vice versa. CF considers the dependency between user ratings, but misses the dependency between item features. CB considers the latter, but not the former. Since hybrid approaches run CB and CF separately, they miss the existed dependency between user ratings and item features. Our model, discloses the duality between user ratings and item features, to reveal the actual reasons of their rating behavior. Moreover, we introduce a scheme to weight features, according to their impact on users preferences. Thus, similarity between users is measured with respect to the dominant features in their profiles.

3 Examined Factors

In this section, we provide details for the examined factors that are involved in CF algorithms. Table 1 summarizes the symbols that are used in the sequel.

Table 1. Symbols and definitions

Symbol	**Definition**	**Symbol**	**Definition**
k	number of nearest neighbors	N	size of recommendation list
P_τ	threshold for positive ratings	$\mathcal{F}_u$	set of features correlated with user u
$\mathcal{U}$	domain of all users	$W(u,f)$	the correlation of user u on feature f
$\mathcal{F}$	domain of all features	$R(u,i)$	the rating of user u on item i
$\mathcal{I}$	domain of all items	W	the weighted user-feature matrix
u, v	some users	R	the user-item ratings matrix
i	some items	P	the user-feature matrix
f	some features	F	the item-feature matrix

Neighborhood size: The number, k, of nearest neighbors used for the neighborhood formation is important because it can affect substantially the system's accuracy. In most related works, k has been examined in the range of values between 10 and 100. The optimum k depends on the data characteristics (e.g., sparsity). Therefore, CF and CB algorithms should be evaluated against varying k, in order to tune it.

Positive rating threshold: It is evident that recommendations should be "positive", as it is not success to recommend an item that will be rated with, e.g., 1 in 1-5 scale. Thus, "negatively" rated items should not contribute to the increase of accuracy. We use a rating-threshold, P_τ, to recommended items whose rating is not less than this value. If we do not use a P_τ value, then the results become misleading.

Train/Test data size: There is a clear dependence between the training set's size and the accuracy of CF and CB algorithms [7]. Therefore, these algorithms should be evaluated against varying training data sizes.

Recommendation list's size: The size, N, of the recommendation list corresponds to a tradeoff: With increasing N, the absolute number of relevant items (i.e., recall) is expected to increase, but their ratio to the total size of the recommendation list (i.e., precision) is expected to decrease. In related work [7], N usually takes values between 10 and 50.

Evaluation Metrics: Several metrics have been used for the evaluation of CF and CB algorithms. We focus on widely accepted metrics from information retrieval. 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).

Notice that with the previous definitions, when an item in the top-N list is not rated at all by the test user, we consider it as *irrelevant* and it counts negatively to precision (as we divide by N). In the following we also use $\mathrm{F}_1 = 2 \cdot \text{recall} \cdot \text{precision}/(\text{recall} + \text{precision})$. F_1 is used because it combines both the previous metrics.

4 Proposed Methodology

The outline of our approach consists of four steps:

1. The content-based user profile construction step: It constructs a content-based user profile from both collaborative and content features.
2. The feature-weighting step: We quantify the affect of each feature inside the user's profile(find important intra-user features) and among the users (find important inter-users features).
3. The formation of user's neighborhood algorithm: To provide recommendations, we create the user's neighborhood, calculating the similarity between each user.
4. The top-N list generation algorithm: We provide for each test user a Top-N recommendation list based on the most frequent features in his neighborhood.

In the following, we analyze each step in detail. To ease the discussion, we will use the running example illustrated in Figure 1a, where I_{1-6} are items and U_{1-4} are users. The null (not rated) cells are presented with dash. Moreover, in Figure 1b, for each item we have four features that describe its characteristics.

4.1 The Content-Based User Profile Construction

We construct a feature profile for a user from both user ratings and item features. In particular, for a user u who rated positively (above P_τ) some items, we build a feature profile to find his favorite features.

In particular, matrix $R(u,i)$ denotes the ratings of user u on each item i. We use a boolean matrix F, where $F(i,f)$ element is one, if item i contains feature

	I_1	I_2	I_3	I_4	I_5	I_6
U_1	-	4	-	-	5	-
U_2	-	3	-	4	-	-
U_3	-	-	-	-	-	4
U_4	5	-	3	-	-	-

(a)

	F_1	F_2	F_3	F_4
I_1	0	1	0	0
I_2	1	1	0	0
I_3	0	1	1	0
I_4	0	1	0	0
I_5	1	1	1	0
I_6	0	0	0	1

(b)

	F_1	F_2	F_3	F_4
U_1	2	2	1	0
U_2	1	2	0	0
U_3	0	0	0	1
U_4	0	2	1	0

(c)

Fig. 1. (a) User-Item matrix R, (b) Boolean Item-Feature matrix F (c) User-Feature matrix P

f and zero otherwise. In our running example, matrices R and F are illustrated in Figures 1a and 1b, respectively. For a user u, his profile is constructed with matrix $P(u,f)$, with elements given as follows:

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

Therefore, $P(u,f)$ denotes the correlation between user u and feature f. Notice that we use only the positively rated items i (i.e., $R(u,i) > P_\tau$) by user u.

In our running example (with $P_\tau = 2$), we construct the P matrix by combining information from R and F matrices. As we can see in Figure 1c, the new matrix P reveals a strong similarity (same feature preferences) between users U_1 and U_4. This similarity could not be derived from the corresponding user ratings in the R matrix.

4.2 The Feature-Weighting of the User Profile

Let $\mathcal{U}$ be the domain of all users and $\mathcal{F}_u$ the set of features that are correlated with user u, i.e., $\mathcal{F}_u = \{f \in \mathcal{F} \mid P(u,f) > 0\}$. Henceforth, user u and feature f are correlated when $P(u,f) > 0$.

We will weight the features of matrix P, in order to find (i) those features which better describe user u (describe the $\mathcal{F}_u$ set) and (ii) those features which better distinguish him from the others (distinguishing him from the remaining users in the $\mathcal{U}$ domain). The first set of features provides quantification of intra-user similarity, while the second set of features provides quantification of inter-user dissimilarity.

In our model, motivated from the information retrieval field and the TFIDF scheme [1], intra-user similarity is quantified by measuring the frequency of each feature f for a user u. Henceforth, this factor is referred as *Feature Frequency* (FF) factor. Furthermore, inter-user dissimilarity is quantified by measuring the inverse of the frequency of a feature f among all users. Henceforth, this factor is referred as *Inverse User Frequency*(IUF) factor.

Thus, *Feature Frequency* $FF(u,f)$ is the number of times feature f occurs in the profile of user u. In our model, it holds that $FF(u,f)=P(u,f)$. The *User*

Frequency $UF(f)$ is the number of users in which feature f occurs at least once. Finally, the *Inverse User Frequency* $IUF(f)$ can be calculated from $UF(f)$ as follows:

$$IUF(f) = \log \frac{|\mathcal{U}|}{UF(f)}. \tag{2}$$

In Equation 2, $|\mathcal{U}|$ is the total number of users. The *Inverse User Frequency* of a feature is low, if it occurs in many users' profiles, whereas it is high, if the feature occurs in few users profiles. Finally, the new weighted value of feature f for user u is calculated as following:

$$W(u, f) = FF(u, f) * IUF(f) \tag{3}$$

This feature weighting scheme represents that a feature f is an important indexing element for user u, if it occurs frequently in it. On the other hand, features which occur in many users' profiles are rated as less important indexing elements due to the low inverse user frequency.

In our running example, the matrix P of Figure 1c is transformed into the matrix W in Figure 2a. As it can be noticed in matrix P, features F_1 and F_2 for user U_1 have the same value, equal to two. In contrast, in matrix W, the same features are weighted differently (0.60 and 0.24). It is obvious now that feature F_1 for user U_1 is an important discriminating feature, whereas this could not be noticed in matrix P.

	F_1	F_2	F_3	F_4
U_1	0.60	0.24	0.30	-
U_2	0.30	0.24	0	-
U_3	-	-	-	0.60
U_4	0	0.24	0.30	-

(a)

	U_1	U_2	U_3	U_4
U_1	-	0.96	0	1
U_2	0.96	-	0	1
U_3	0	0	-	0
U_4	1	1	0	-

(b)

Fig. 2. (a) weighted User-Feature matrix W (b) User-User similarity matrix

4.3 The User's Neighborhood Formation

To provide recommendations, we need to find similar users. In our model, as it is expressed by equation 4, we apply cosine similarity in the weighted user-feature W matrix. We adapt cosine similarity to take into account only the set of features, that are correlated with both users. Thus, in our model the similarity between two users is measured as follows:

$$\mathrm{sim}(u, v) = \frac{\sum\limits_{\forall f \in X} W(u, f) W(v, f)}{\sqrt{\sum\limits_{\forall f \in X} W(u, f)^2} \sqrt{\sum\limits_{\forall f \in X} W(v, f)^2}}, X = \mathcal{F}_u \cap \mathcal{F}_v. \tag{4}$$

In our running example, we create a user-user matrix according to equation 4, where we can find the neighbors of each user (those which have the higher value, are the nearest ones). In Figure 2b, we can see that the nearest neighbor of user U_2 is U_4 with similarity 1, and U_1 follows with similarity value 0.96.

4.4 The Top-N List Generation

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.

In our running example, assuming that we recommend a $top-1$ list for U_2 (with k=2 nearest neighbors), we work as follows:

1. We get the nearest neighbors of U_2: $\{U_4, U_1\}$
2. We get the items in the neighborhood: $\{I_1, I_3, I_5\}$
3. We get the features of each item: I_1: $\{F_2\}$, I_3: $\{F_2, F_3\}$, I_5: $\{F_1, F_2, F_3\}$
4. We find their frequency in the neighborhood: $fr(F_1)$=1, $fr(F_2)$=3, $fr(F_3)$=2
5. For each item, we add its features frequency finding its weight in the neighborhood: $w(I_1) = 3$, $w(I_3) = 5$, $w(I_5) = 6$.

Thus, I_5 is recommended, meaning that it consists of features that are prevalent in the feature profiles of U_2's neighbors.

5 Performance Study

In this section, we study the performance of our feature-weighted user model against the well-known CF, CB and a hybrid algorithm, by means of a thorough experimental evaluation. For the experiments, the Featured-Weighted User Model is denoted as FWUM, the collaborative filtering algorithm as CF and the content-based algorithm as CB. Finally, as representative of the hybrid algorithms, we have implemented a state-of-the-art algorithm, the Cinemascreen Recommender Agent [6], 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%). The metrics we use are recall, precision, and F_1. P_τ threshold is set to 3. Finally, we consider the division between not hidden and hidden data. 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). In the imdb database there are 28 different movie genres (Action, Film-Noir, Western etc.), 32882 different keywords referring to movie characteristics, 121821 directors and 1182476 actors and actresses (a movie can be classified to more genres or keywords). We joined the aforementioned data with one real data set that has been used as benchmark in prior work. In particular, we used the 100K MovieLens [5] data set with 100,000 ratings assigned by 943 users on 1,682 movies. The range of ratings is between 1(bad)-5(excellent) of the numerical scale. The joining process lead to 23 different genres, 9847 keywords, 1050 directors and 2640 different actors and actresses (we selected only the 3 most paid actors or actresses for each movie). Finally, notice that we have validated the presented results with other real data sets (Movielens 1M and EachMovie). Due to lack of space, these results will be presented in a extended version of this work.

5.1 Comparative Results for CF Algorithms

Firstly, we compare the two main CF algorithms, denoted as user-based (UB) and item-based (IB) algorithms. The basic difference between these two CF algorithms is that, 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 R matrix). Figure 3 demonstrates that item-based CF compares favorably against user-based CF 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.

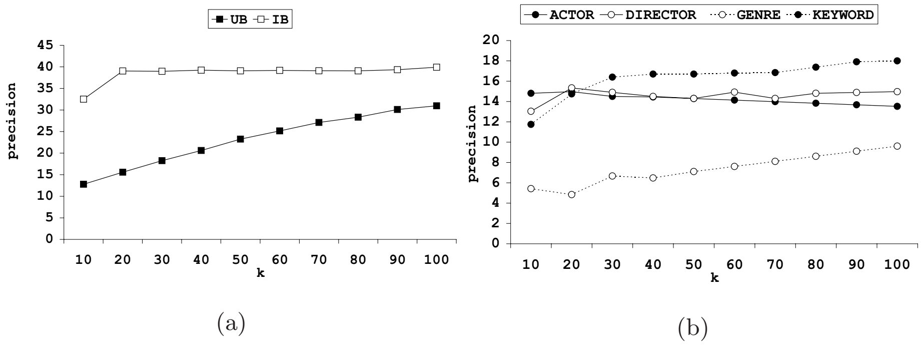

Fig. 3. Comparison in terms of precision between (a) CF algorithms (b) CB classes of features

5.2 Comparative Results of Feature Classes for CB Algorithm

As it is already discussed, 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. Pure CB algorithm exploits information derived only from document or item features. Thus, we create an item-item similarity matrix based on cosine similarity applied on features of items (by exploiting information only from the item-feature F matrix). In Figure 3b, 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.

5.3 Comparative Results for CF, CB, CFCB and FWUM Algorithms

We test the FWUM algorithm against CF, CB and CFCB algorithms using the best options as they have resulted from the previous measurements. In Figures 4a and 4b, we see results for precision and recall. FWUM presents the best performance in terms of precision (above 60%) and recall(above 20%). The reason is two-fold:(i) the sparsity has been downsized through the features and (ii) the applied weighting-schema reveals the actual user preferences.

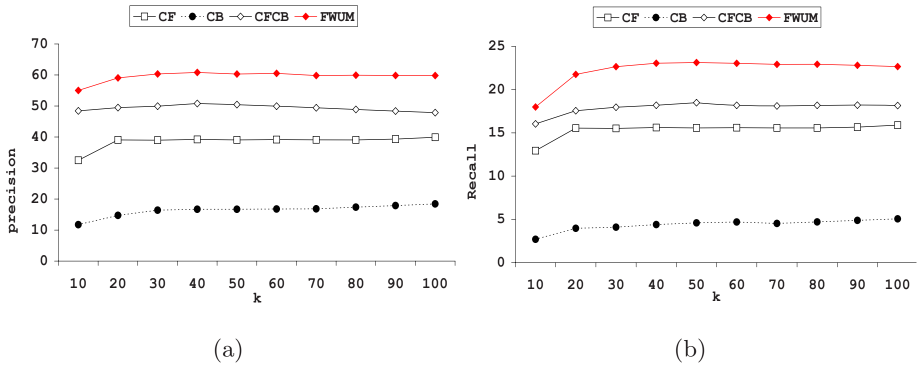

Fig. 4. Comparison between CF, CB and CFCB with FWUM in terms of (a) precision (b) recall

5.4 Examination of Additional Factors

Recommendation list's size: We examine the impact of N. The results of our experiments are depicted in Figures 5a and 5b. As expected, with increasing N, recall increases and precision decreases. Notice that the FWUM outperforms CF, CB and CFCB in all cases. The relative differences between the algorithms are coherent with those in our previous measurements.

Training/Test data size: Now we test the impact of the size of the training set. The results for the F_1 metric are given in Figure 5c. As expected, when the training set is small, performance downgrades for all algorithms. Similar to the previous measurements, in all cases FWUM algorithm is better than CF, CB and CFCB cases and low training set sizes do not affect determinatively the FWUM accuracy.

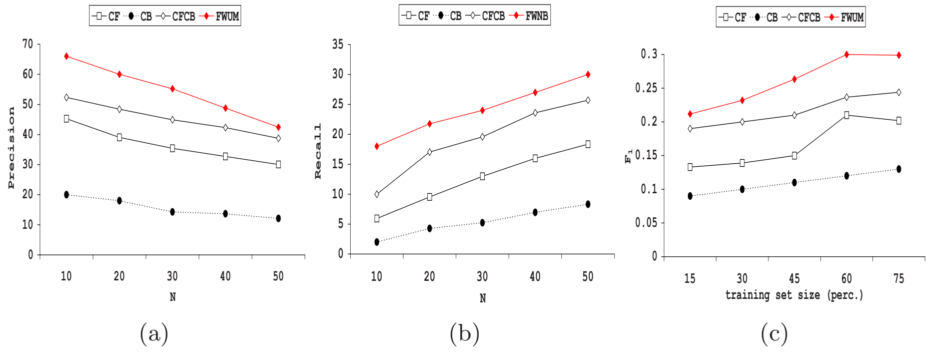

Fig. 5. Comparison vs.: (a) N precision, (b) N recall, (c) training set size

6 Conclusions

We proposed a feature-weighted user model for recommender systems. We perform experimental comparison of our method against well known CF, CB and a hybrid algorithm with a real data set. Our approach shows significant improvements in accuracy of recommendations over existing algorithms. The reason is that our approach reveals the favorite features of a user and recommends those items that are composed of these features. Summarizing the aforementioned conclusions, our feature-weighted user model is a promising approach for getting more robust recommender systems. In our future work, we will consider the fusion of different classes of features in a multivariate user model for getting more precise recommendations.

References

1. Baeza-Yates, R.A., Ribeiro-Neto, B.A.: Modern Information Retrieval. ACM Press / Addison-Wesley (1999)
2. Balabanovic, M., Y, S.: Fab: Content-based, collaborative recommendation. ACM Communications 40(3), 66–72 (1997)
3. Jin, X., Zhou, Y., Mobasher, B.: A maximum entropy web recommendation system: Combining collaborative and content features. In: Proc. ACM SIGKDD Conf., pp. 612–617 (2005)
4. Melville, P., Mooney, R.J., Nagarajan, R.: In: Proc. AAAI conf. In Content-Boosted Collaborative Filtering for improved Recommendations, pp. 187–192 (2002)
5. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J.: Grouplens: An open architecture for collaborative filtering on netnews. In: Proc. Conf. Computer Supported Collaborative Work, pp. 175–186 (1994)
6. Salter, J., Antonopoulos, N.: Cinemascreen recommender agent: Combining collaborative and content-based filtering. Intelligent Systems Magazine 21(1), 35–41 (2006)
7. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: Proc. WWW Conf., pp. 285–295 (2001)
8. Yan, W.T., Molina, H.G.: Sift: A tool for wide-area information dissemination. In: Proc. UNSENIX Conf., pp. 177–186 (1995)