EXPLAINING RECOMMENDATIONS OF MOVIES IN WEB VIDEO RENTAL BUSINESSES

Panagiotis Symeonidis[1] and Yannis Manolopoulos[2]

Abstract

Have you ever watched a movie that somebody recommended to you, and felt you wasted your time? If yes, for sure you have wished that there was a way to get, along with the recommendation, the reasoning behind it. Providing explanations along with the recommendations is the solution. Our prototype system MoviExplain is a movie recommender system with robust explanations. MoviExplain proposes a novel approach that attains both accurate and justifiable recommendations. It constructs a feature profile for the users, to reveal their favorite features. We group users into biclusters (i.e., groups of users which exhibit highly correlated ratings on groups of movies) to exploit partial matching between the preferences of the target user and each community of users.

1. Introduction

Recent research noticed that the acceptance of Collaborative Filtering (CF) recommender systems (like Amazon.com, MovieLens etc.) increases, when users receive justified recommendations [6]. For instance, Amazon adopted the following two styles of justification: (i) "Customers who bought item X also bought items $Y, Z, \ldots$". This is the so called "nearest neighbor" style [3] of justification. (ii) "Item Y is recommended because you rated item X and more". This is the so called "influence" style, where the system isolates the item, X, that influenced most the recommendation of movie Y.

Pure Content-Based filtering (CB) systems [9] make recommendations for a target user based on the past data of that user without involving data from other users. Based on pure CB, several research works [4, 9] were able to provide explanations for their recommendations. For instance, Billsus and Pazzani [4] recommend news articles to users, providing the following style of justification. "This story received a high relevance score, because it contains the words f_1, f_2, and f_3". This is a template of the so called "keyword" style [3] of justification.

Bilgic et al. [3] claimed that the "influence" and "keyword" styles are better than the "nearest neighbor" style, because they allow users to accurately predict their true opinion of an item. Nevertheless, both "influence" and "keyword" styles can not justify adequately their recommendations, because they are based solely either on data about ratings (rating data), or solely on *content* data, which are extracted in the form of features that are derived from the items.

Several CF systems have proposed the combination of content data with rating data [2, 8, 11] to tackle: (i) the possible unwillingness of users to provide ratings or (ii) the new users in the system which have not submitted any ratings yet (i.e., the Cold-Start problem). Both these facts result to data sparsity. By

[1]Department of Informatics, Aristotle University, Thessaloniki, Greece email: symeon@csd.auth.gr
[2]Department of Informatics, Aristotle University, Thessaloniki, Greece email: manolopo@csd.auth.gr

combining CF with CB, data sparsity can be reduced, yielding to more accurate recommendations. For this reason, recently proposed recommender systems, like CinemaScreen [11] and Libra [3], combine CB and CF in their recommendations. Nevertheless, although these recommender systems provide accurate recommendations, they cannot adequately justify them.

Our prototype system MoviExplain is a movie recommender system with explanations that provides solutions to the aforementioned problems. It relies on the democratic nature of voting. In essence, MoviExplain uses a simple heuristic to interpret a rating by a user A to a movie B, as a vote to the features of movie B (actors, directors etc.). Based on these features, MoviExplain builds a feature profile for each user.

MoviExplain groups users into *biclusters*, i.e., group of users which exhibit highly correlated ratings on groups of movies, to detect partial matching of user's preferences. Each bicluster acts like a community for its corresponding movies; e.g., in a system that recommends movies, such a group may be users that prefer comedies. Moreover, by using groups instead of individual users, the extracted features are collective, reflecting preferences of whole communities. As a result, collective features cover wider range of users preferences and result to better explanations.

The justification style of MoviExplain combines "keyword" with "influence" explanation styles [3], having the following form: "Movie X is recommended because it contains features $a, b, \ldots$ which are also included in movies $Z, W, \ldots$ you have already rated". If inside the user's feature profile, these features are frequent, this is a strong evidence for justifying the recommendations.

2. Related Work

Regarding research on explanations, many pure CB systems have tried to provide explanations to users. For instance, Billsus and Pazzani [4] recommend news articles to users, providing also explanations for reasoning their recommendations. In 2000, Mooney and Roy [9] proposed a method based also on pure CB for recommending books. These works were pioneering for the problem of explanation and inspired subsequent research on combining CF and CB for explanation purposes. In the area of CF, there is a little existing research on explaining. In 2000, Herlocker et al. [6] proposed 21 different interfaces of explaining CF recommendations. By conducting a survey, they claim that the "nearest neighbor" style is effective in supporting explanations. Amazon.com's recommender system early adopted the "nearest neighbor" explanation style. In 2005, Bilgic et al. [3] demonstrate, through a survey, that the "influence" and "keyword" styles are better than the "nearest neighbor" style, because they help users to accurately predict their true opinion of a recommendation.

3. MoviExplain Web Crawler

MoviExplain uses a web crawler to search for information about movies on the Web. The movies information concerns the basic movies characteristics like its cast (directors and actors), their official web pages, posters and various photos, movie genres etc. Moreover, a search engine summarizes this content and adds the appropriate links to their indexes. Thus, a user can search for his favorite movie using the MoviExplain search engine and get updated information about its features. MoviExplain is fully integrated to the well-known Internet Movie Data Base (IMDB) web site.

4. MoviExplain Database Profiles

As described previously, MoviExplain's database profiles contain users ratings and movies features. The feature extraction has been done from the Internet Movie Database (IMDB). In this work, following related research, e.g. [11], we select as movies features the actors, directors, and genres. In the following, we describe how we build the feature profile of a user.

CF algorithms are based solely on rating data. We assume that the rating data are given with matrix R, where the rating of a user u over a movie m is given from the element $R(u, m)$. An example is given in Figure 2a, where M_{1-7} are movies and U_{1-8} are users. The null cells (no rating) are presented with dash. As rating profile $R(U_k)$ of user U_k, we define the set of rated (i.e., non null) movies in the k-th row of matrix R. For instance, $R(U_1) = \{M_1, M_3, M_5\}$.

In contrast, CB algorithms are based solely on content data. We assume matrix F, where $F(m,f)$ element is one, if movie m contains feature f and zero otherwise. In Figure 2b, for each movie we have four features that describe its characteristics. For example, these features can be names of actors or directors. As movie profile $F(M_k)$ of movie M_k, we define the set of features f for which $F(M_k, f)$ have one value. For instance, $F(M_2) = \{f_2, f_3\}$.

	M_1	M_2	M_3	M_4	M_5	M_6	M_7
U_1	5	-	2	-	1	-	-
U_2	2	-	4	1	4	3	-
U_3	4	-	2	-	2	-	5
U_4	-	3	1	4	-	5	2
U_5	-	2	4	2	5	1	-
U_6	5	1	-	1	-	-	3
U_7	-	2	5	-	4	1	-
U_8	1	4	-	5	4	3	-

(a)

	f_1	f_2	f_3	f_4
M_1	1	0	0	0
M_2	0	1	1	0
M_3	0	0	1	1
M_4	0	1	0	1
M_5	0	1	1	1
M_6	1	0	1	1
M_7	1	0	1	0

(b)

	f_1	f_2	f_3	f_4
U_1	1	0	0	0
U_2	1	1	3	3
U_3	2	0	1	0
U_4	1	2	2	2
U_5	0	1	2	2
U_6	2	0	1	0
U_7	0	1	2	2
U_8	1	3	3	3

(c)

Figure 1. Running example: (a) User-movie matrix R (b) Movie-Feature matrix F (c) User-Feature matrix P

For the construction of the feature profile of a user, we use a positive rating threshold, P_τ, to select the items whose rating express a "positive" preference by the user. Thus, to find the favorite features (actors, directors etc.) of a user u who rated positively (above P_τ) some movies, we build his feature profile. We assume a matrix, P, whose element $P(u, f)$ denotes the correlation between user u and feature f, and is given by Equation 1.

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

In our running example, assuming that $P_\tau = 2$, the resulting P matrix is depicted in Figure 2c. As feature profile $P(U_k)$ of user U_k, we define the set of $P(u, f)$ elements in k-th row of matrix P with value higher than zero. For instance, $P(U_5) = \{(f_2, 1), (f_3, 2), (f_4, 2)\}$.

5. MoviExplain Recommendation Engine

The Recommendation Engine is the heart of the MoviExplain system. It aims to provide both accurate and justifiable recommendations. The recommendation algorithm contains four stages: (i) The creation of user groups , (ii) the feature-weighting, (iii) the neighborhood formation, and (iv) the generation of the recommendation and justification lists. In the following, we describe each stage in detail.

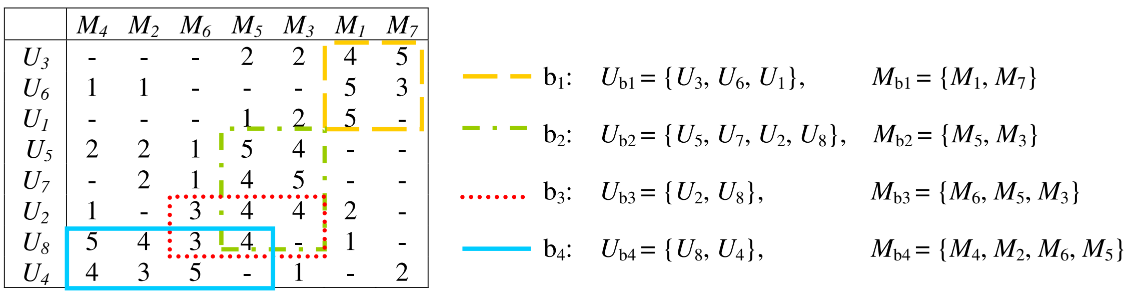
5.1. Creation of Users Groups

To provide recommendations to a target user u, we have to find groups of users that have similar rating behavior as u. Therefore, users have to be organized into groups according to their ratings. We propose the simultaneous clustering of users and movies (biclustering), which discovers groups of users exhibiting highly correlated ratings on groups of movies. The creation of biclusters is done off-line, without burdening the on-line recommendation procedure. This means that our approach can provide real-time recommendations very efficiently in terms of execution time.

In MoviExplain, biclusters allow the computation of similarity between u and a bicluster b *only* on the items or features that are included in b. Thus, partial matching of preferences is taken into account. Moreover, we are able to extract collective community features that can be used for justifying our recommendations. In particular, we adopt the xMotif algorithm, which looks for subsets of rows and subsets of columns with coherent values [10].

A bicluster b, detected by xMotif, corresponds to a subset of users U_b that jointly present coherent rating behavior across a subset of movies M_b, and satisfies the following conditions: (i) the number of users in U_b is a fraction (user-defined) of all users, (ii) every movie in M_b is conserved across all the users in U_b, and (iii) for every movie not in M_b, the movie is conserved in at most a fraction (user-defined) of the users in U_b. These conditions mean that xMotif allows a user or movie to be included in a bicluster, even if there exist a fraction of their rating values which cannot not defined as interesting ones (ratings under the P_τ threshold or null values). This is useful for the case of sparse rating data.

In Figure 3, we have applied the xMotif algorithm in our running example. We found four biclusters which consist, at least, of 2 users and 2 movies.



	M_4	M_2	M_6	M_5	M_3	M_1	M_7
U_3	-	-	-	2	2	4	5
U_6	1	1	-	-	-	5	3
U_1	-	-	-	1	2	5	-
U_5	2	2	1	5	4	-	-
U_7	-	2	1	4	5	-	-
U_2	1	-	3	4	4	2	-
U_8	5	4	3	4	-	1	-
U_4	4	3	5	-	1	-	2

b1: $U_{b1} = \{U_3, U_6, U_1\}$, $M_{b1} = \{M_1, M_7\}$

b2: $U_{b2} = \{U_5, U_7, U_2, U_8\}$, $M_{b2} = \{M_5, M_3\}$

b3: $U_{b3} = \{U_2, U_8\}$, $M_{b3} = \{M_6, M_5, M_3\}$

b4: $U_{b4} = \{U_8, U_4\}$, $M_{b4} = \{M_4, M_2, M_6, M_5\}$

Figure 2. xMotif algorithm is applied to matrix R finding 4 biclusters in our running example.

Notice that two biclusters may overlap, which means that several rows or columns of the matrix may participate in multiple biclusters. Moreover, to understand how xMotif finds biclusters with coherent values, notice that in bicluster b_4, movie M_5 is included, even if, it has not been rated by user U_4.

5.2. Feature-weighting

In this step, we weight the feature profiles of users, to find (i) those features which better describe a user and (ii) those features which better distinguish him from others. The feature-weighting scheme is based on the well-known TFIDF scheme [1]. Thus, from P matrix we define a new weighted matrix W. As mentioned, we use a simple heuristic to interpret a rating by a user A to a movie B, as a vote to all features (actors, directors etc.) of B. Therefore, feature-weighting compensates for the cases where A does not necessarily likes all the features of the movie, because weighting designates only the features that are relevant to A.

Finally, similarly to the users' weighting scheme, we weight the features contained in each found bicluster b. In particular, similarly to the feature profile of users, we generate a feature profile of each bicluster. We define P_B matrix whose elements $P_B(b, f)$ are given from the frequency of feature f in a bicluster b. Thus, from P_B matrix we generate the weighted W_B matrix.

5.3. Neighborhood Formation

In the following, we search the groups (biclusters) we have formed to find the k nearest ones to u. The neighborhood formation requires to measure the similarity of u to each of the biclusters.

To measure similarity between a user u and a bicluster b, we have two options:

(i) To consider *only* the similarity between the set of movies M_b that are included in b and the set of movies $\mathcal{M}_u$ rated by u. In Equation 2, we adapt the cosine similarity measure to consider the aforementioned option:

$$\mathrm{sim}_I(u,b) = \frac{\sum\limits_{\forall m \in \mathcal{M}_u \cap \mathcal{M}_b} R(u,m) R_B(b,m)}{\sqrt{\sum\limits_{\forall m \in \mathcal{M}_u \cap \mathcal{M}_b} R(u,m)^2} \sqrt{\sum\limits_{\forall m \in \mathcal{M}_u \cap \mathcal{M}_b} R_B(b,m)^2}} \tag{2}$$

This way, u is partially matched with b, by focusing only on the subset of movies $M_u \cap M_b$ and ignoring the rest movies that u has rated. Note that u can be matched against several biclusters, thus each bicluster contributes as candidates for recommendation, the movies it specializes on.

(ii) To consider *only* the similarity the set of features F_b that are included in b and the set of features $\mathcal{F}_u$ of u (these are the features of u given in the weighted matrix W). In Equation 3, we adapt the cosine similarity to consider this option:

$$\mathrm{sim}_F(u,b) = \frac{\sum\limits_{\forall f \in \mathcal{F}_u \cap \mathcal{F}_b} W(u,f) W_B(b,f)}{\sqrt{\sum\limits_{\forall f \in \mathcal{F}_u \cap \mathcal{F}_b} W(u,f)^2} \sqrt{\sum\limits_{\forall f \in \mathcal{F}_u \cap \mathcal{F}_b} W_B(b,f)^2}} \tag{3}$$

As our objective is to provide both accurate and justifiable recommendations, we weight equally (with weight 0.5) the two similarity measures into a single one, which is given in Equation 4:

$$\mathrm{sim}(u,b) = 0.5 * \mathrm{sim}_I(u,b) + 0.5 * \mathrm{sim}_F(u,b) \tag{4}$$

In our running example, for U_1 and number of neighbors $k = 2$, the nearest-biclusters of U_1 are $\langle b_1, b_4 \rangle$.

5.4. Generation of the Recommendation and Justification Lists

For the target user u, the final step is to generate the recommended movies and the justification for each recommended movie. We identify the movies in u's bicluster neighborhood, which are both: (i) highly preferred by other users, according to their ratings in the R_B matrix, and (ii) contain significant features, according to the weighed matrix W_B (we exclude movies that have been already rated by u). By taking into account both factors, we sort the movies and recommend the N top ones (N is a parameter).

In our running example, assume that we want to recommend one movie to user U_1 (thus, $N = 1$). By using only the first nearest bicluster of U_1 (i.e., $k = 1$), which is b_1, we get the features of the movies in b_1. The movies in b_1 are $M_{b_1} = \{M_1, M_7\}$. The feature profile of M_1 is $F(M_1) = \{f_1\}$ and the feature profile of M_7 is $F(M_7) = \{f_1, f_3\}$. Next, we find the frequency of the features inside the neighborhood of U_1. We get that, $fr(f_1) = 2$ and $fr(f_3) = 1$. Finally, for each movie, we sum the frequency of its features and find its weight: $w(M_1) = 1$, $w(M_7) = 3$. Thus, M_7 is recommended, because it has higher weight than M_1.

The justification for each recommended movie is generated by those features in the movie's profile, which exist also in u's feature profile. The qualifying features are ordered according to their frequency in the u's feature profile, and form the justification list for the recommended movie.

In our running example, f_1 is the only feature existing both in the profile of the recommended movie M_7 and the feature profile of the target user U_1. Therefore, the justification that U_1 receives for M_7 will be the following: "Movie M_7 is recommended, because it contains feature f_1, which is included also in movie M_1 you have already rated".

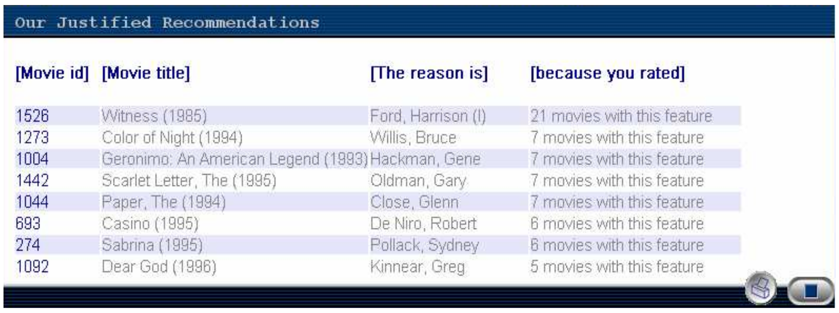
6. MoviExplain Web Site

Users interact with MoviExplain through its web site http://delab.csd.auth.gr/MoviExplain.

MoviExplain consists of 3 sub-systems: (i) the Search Engine, (ii) the Rating System and (iii) the Recommendation with Explanation System. The Search Engine keeps updated information about movies and their features, which are collected by the web-crawler. The Rating System is meant to help a user to keep track of the movies he has rated. Based on these features, MoviExplain builds a feature profile for each user. Finally, MoviExplain provides as explanation, the feature that influenced most a recommendation, showing also how strong is this feature in the feature profile of a user. As shown in Figure 4, the link "The reason is" reveals the favorite feature that influenced most the MoviExplain's recommendations, while the link "because you rated" shows how strong is this feature in the feature profile of a user.

7. Experimental Results

In this section, we experimentally study the performance of the proposed MoviExplain System. For comparison purposes, we include as representative of the hybrid CFCB algorithms, the CinemaScreen Recommender Agent [11] denoted as CinemaScreen. As representative of the hybrid CBCF algo-


Our Justified Recommendations

[Movie id]	[Movie title]	[The reason is]	[because you rated]
1526	Witness (1985)	Ford, Harrison (I)	21 movies with this feature
1273	Color of Night (1994)	Willis, Bruce	7 movies with this feature
1004	Geronimo: An American Legend (1993)	Hackman, Gene	7 movies with this feature
1442	Scarlet Letter, The (1995)	Oldman, Gary	7 movies with this feature
1044	Paper, The (1994)	Close, Glenn	7 movies with this feature
693	Casino (1995)	De Niro, Robert	6 movies with this feature
274	Sabrina (1995)	Pollack, Sydney	6 movies with this feature
1092	Dear God (1996)	Kinnear, Greg	5 movies with this feature

Figure 3. Explaining Recommendations

rithms, we use the Libra System [3] denoted as Libra. Finally, we include in our experiments a state-of-the-art cluster-based CF algorithm [7] denoted as DM. We compare all approaches in terms of their recommendations accuracy and their ability to provide qualitative explanations, by using an objective measure.

Our experiments are performed with the 100K MovieLens real data set, which consists of 100,000 ratings assigned by 943 users on 1,682 movies. The range of ratings is between 1(bad)-5(excellent). The extraction of the content features has been done by joining with the contents of the internet movie database (imdb) and selecting 3 different classes of features: genres, actors, and directors. The join process yielded 23 different genres, 1,050 directors and 2,640 different actors and actresses. To ease real users recognize the features used for explanation, we focused on the 3 most paid actors or actresses as features of each movie, because they are often well-known.

In the following experiments, the default size of the recommendation list, N, is set to 20, the neighborhood size k, is set to 10, the size of the training set is set to 75% and the threshold for positive rating P_τ is set to 3. For the biclusters generations, xMotif algorithm resulted in 991 biclusters.

7.1. Evaluating Recommendations and Explanations

A recommender system provides to a user, u, list $L = \{M_1, \ldots, M_N\}$ with the N top recommended items. To measure the accuracy of the recommendation list L, we use the well known measures of precision and recall ([11] argues that, due to the particularities of movie industry, precision may be more significant than recall). In particular: N denotes the size of the recommendation list L, R_L denotes the number of relevant items (i.e., those rated higher than P_τ by u) that are included in L, and R denotes the total number of relevant items for u. Precision and recall are defined as follows:

- *Precision* is the ratio of R_L to N.
- *Recall* is the ratio of R_L to R.

Precision and recall concern only the rating profile of a user u and measure the accuracy of L. However, precision and recall cannot distinguish between a relevant item from a more relevant item [1]. To cope with this problem and to measure the quality of the justification, we introduce a user-oriented measure, called *explain coverage*. Since we are interested in the overall quality of the provided justifications, for a user u that receives a recommendation list L, we combine the justifications of all recommended movies in a single, ordered *justification* list $J = \{(f_1, c_{f_1}), \ldots, (f_m, c_{f_m})\}$. This way,

J contains the union of features that are included in all separate justifications of movies in L. Since a feature may appear in the justification of more than one movies, for each feature in J we sum the corresponding frequencies in all separate justifications. Thus, each pair (f_i, c_{f_i}) denotes that feature f_i has overall frequency c_{f_i} inside L. In our running example, by assuming that $N = 2$ movies are recommended, we can find that these are movies M_7 and M_1. Since only feature f_1 exists in the profiles of M_7 and M_1 and U_1 (the target user), we have that $J = \{(f_1, 2)\}$.

For a user u that receives a recommendation list L, the *explain coverage* for the justification list J is defined as follows:

$$Explain\ \ coverage(u, J) = \frac{\sum\limits_{\forall (f_i, c_{f_i}) \in J} \min\{c_{f_i}, P(u, f_i)\}}{\sum\limits_{\forall f_i \in F} P(u, f_i)} \tag{5}$$

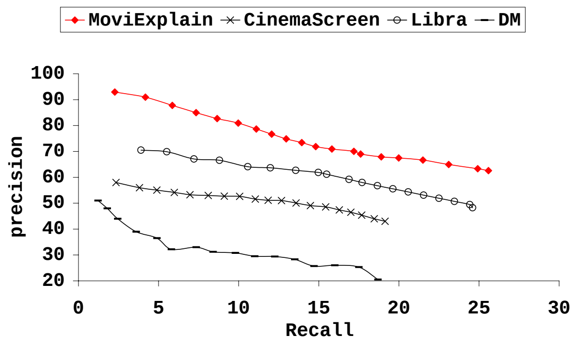
Explain coverage takes values in the range [0, 1], whereas values closer to 1 correspond to better coverage. In our running example, we have previously found that $L = \{M_1, M_7\}$ and $J = \{f_1, 2\}$. Since $P(U_1) = \{f_1\}$, we get *coverage* equal to $\min\{2, 1\}/1 = 1$. *Explain coverage* measures how much the features in u's feature profile are covered by the features of the items included in L. If *explain coverage* is high, then the justification is more effective, as the features that are included in the justification list J can be easily recognized and accepted by the user. Since J contains the union of features in the separate justifications of recommended movies, it characterizes their overall quality.

We also use other coverage metrics to indicate if our system can produce recommendations that cover a wide range of movies. This means that a system should not only provide as recommendations only blockbusters movies but also movies that the user may not be aware of. Thus, we use three additional coverage measures, which have been proposed by Salter and Antonopoulos [11] paper. In particular, the *standard coverage*, the *catalog coverage* and the *prediction coverage* are described as follows:

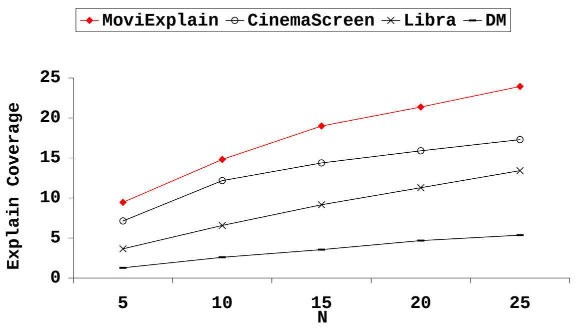
- *Standard coverage* is the average number of movies for which the system produces recommendations for each user. This is calculated as a percentage of the total movies in the database.
- *Catalog coverage* is the percentage of movies for which the system ever produced a recommendation.
- *Prediction coverage* counts how many of the movies that had been rated by the user above P_τ and in our experiments have been removed from his rating profile (to test for unrated/new movies), are finally recommended to him. This is calculated as a percentage of the total number of positive ratings removed from the test data set.

7.2. Measuring Precision, Recall, and Explain Coverage

First, we compare the four algorithms by measuring precision vs. recall. Figure 5a plots the precision-recall diagram for the four algorithms (precision and recall are given as percentages). In particular, to obtain varying precision-recall values, we varied the number of the recommended movies (i.e., the parameter N). As expected, MoviExplain attains the best precision in all cases. The reason is two-fold, MoviExplain takes into account the duality between users and items by using biclustering, and, moveover, it detects partial matching of users' preferences. Libra has better precision than Cinemascreen because its CF algorithmic step is applied after CB, focusing more in the users' ratings. Finally, DM has the smaller precision, because it is not hybrid and does not tackle well the sparsity problem.



(a)



(b)

Figure 4. Comparison between MoviExplain, CinemaScreen, Libra and DM in terms of (a) precision vs. recall and (b) *explain coverage* vs. N.

Next, we compare the four approaches in terms of *explain coverage* vs. the size N of the recommendation list. The results are presented in Figure 5b (*explain coverage* is given as percentage). For all methods, *explain coverage* increases with increasing N, because more recommended movies (resulting from higher N values) are able to cover more the profile of the target users. However, MoviExplain outperforms the other methods in all cases. The reason is that MoviExplain uses groups of users, whereas the other methods are based on individual users. Thus, MoviExplain can extract collective features that provide more qualitative justifications. CinemaScreen has better *explain coverage* than Libra, because its CB algorithmic step is applied after CF, focusing more in the movies' features. DM has the smaller *explain coverage*, because it does not focus on movies' features at all.

7.3. Measuring Recommendations' Diversity

In this Section, we compare the four approaches in terms of the *standard coverage*, the *catalog coverage* and the *prediction coverage*, as shown in Table 1. As expected, *standard coverage* is high for all methods except DM. This is because DM is a pure CF algorithm and can only make recommendations for movies that at least few users have rated. The same applies for the *catalog coverage*. *Catalog coverage* is a good guide to the novelty of recommendations. As shown in column 3 of Table 1, the *catalog coverage* is high for all methods that combine CF and CB, which attain high recommendation diversity. Finally, the *prediction coverage* is very high for all four methods, as shown in column 4 of Table 1. That is, all recommendation techniques are capable of recommending movies that were in the users' rating profile.

8. Conclusions

The need of providing justifiable recommendations has recently attracted significant attention, especially in e-commerce sites (Amazon, e-bay etc.). In this paper, we proposed MovieExplain, a movie recommender system that goes far beyond just recommending movies. It attains both accurate and

Algorithms	*standard coverage*	*catalog coverage*	*prediction coverage*
MoviExplain	**79.7**	85.4	**96.5**
CinemaScreen	64.1	77.1	91.7
Libra	75.2	**86.8**	93.2
DM	33.7	38.7	95.5

Table 1. Comparison between MoviExplain, CinemaScreen, Libra and DM in terms of the *standard coverage*, the *catalog coverage* and the *prediction coverage*

justifiable recommendations, giving the ability to a user, to check the reasoning behind a recommendation. We defined the *explain coverage* ratio as an objective metric to measure the quality of justifications. We performed experiments that illustrate the superiority of our approach over recently proposed hybrid approaches. In the future, we indent to use in MoviExplain also natural language processing to provide more robust explanations.

References

[1] R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.

[2] M. Balabanovic and S. Y. Fab: Content-based, collaborative recommendation. *ACM Communications*, 40(3):66–72, 1997.

[3] M. Bilgic and R. Mooney. Explaining recommendations: Satisfaction vs. promotion. In *Proc. Recommender Systems Workshop (IUI Conf.)*, 2005.

[4] D. Billsus and M. Pazzani. A personal news agent that talks, learns and explains. In *Proc. Autonomous Agents conf.*, pages 268–275, 1999.

[5] T. George and S. Merugu. A scalable collaborative filtering framework based on co-clustering. In *Proceedings of the IEEE ICDM Conference*, 2005.

[6] J. Herlocker, J. Konstan, and J. Riedl. Explaining collaborative filtering recommendations. In *Computer Supported Cooperative Work*, pages 241–250, 2000.

[7] R. Jin, L. Si, and C. Zhai. A study of mixture models for collaborative filtering. *Information Retrieval*, 9(3):357–382, 2006.

[8] P. Melville, R. J. Mooney, and N. R. Content-boosted collaborative filtering for improved recommendations. In *Proc. AAAI conf.*, pages 187–192, 2002.

[9] R. Mooney and L. Roy. Content-based book recommending using learning for text categorization. In *Proc. ACM DL Conf.*, pages 195–204, 2000.

[10] T. Murali and S. Kasif. Extracting conserved gene expression motifs from gene expression data. In *Proceedings of the Pacific Symposim on Biocompomputing Conference*, volume 8, pages 77–88, 2003.

[11] J. Salter and N. Antonopoulos. Cinemascreen recommender agent: Combining collaborative and content-based filtering. *Intelligent Systems Magazine*, 21(1):35–41, 2006.