A Content-Based User Model Generation and Optimization Approach for Movie Recommendation

Oznur KIRMEMIS, Aysenur BIRTURK

Department of Computer Engineering, METU, Ankara, TURKEY
0090 312 210 20 80
{e120321, birturk}@metu.edu.tr

Abstract

Personalization has become a powerful approach for constructing more precise and easy to use information search and recommendation systems. The quality of the personalization is heavily dependent on the accuracy of the user models created by the system and it is very important to incorporate content information of the working domain in order to enrich these models. This paper proposes a content based movie recommendation algorithm to make recommendations for the target user through building content based user models from collaborative-based user models and characteristics of the movie domain. Constructed user models are fine-tuned through "highly liked", "highly not liked", and "don't care" flags. The user models are presented to the users in terms of the most important features and dimensions in their profile. This makes explicit the users' implicit and unknown preferences of the movie domain. The system is evaluated and the results are presented using decision-support metrics.

Introduction

Recommendation systems have become an important technology for helping users to understand and navigate complex product spaces. Most recommendation systems use three approaches for building a user profile and computing recommendations (Adomavicius and Tuzhilin 2005): collaborative filtering, content-based filtering, and hybrid approaches. Collaborative filtering recommends items to a user by matching the user's taste to that of other users in the system. On the other hand, content-based systems recommend items based on the content of the item rather than other users' ratings. Hybrid approaches exploit both content-based and collaborative filtering facilities.

In this paper, we propose a framework for a content-based film recommendation system, OpenMore. We have focused on enhancing user models from collaborative filtering recommenders where a vector of explicit ratings on a set of objects is provided by the user (Herlocker et al. 1999) to the content based user model, represented as a list of preferences (Morita and Shinoda 1994) of different movie features. In (Berkovsky, Kuflik, and Ricci 2006) content-based user models are constructed from collaborative-based user models. We also build content-based user models from collaborative-based user models but with a different construction and optimization approach. We do not use collaborative filtering in the sense of finding the like-minded users; however, we use collaborative-based user models in order to find best values of the defined thresholds that are used for the optimization of the content-based user models. The proposed approach constructs user models that facilitate the identification of commonalities in positively or negatively rated objects as derived by the collaborative user models. Movie profiles are kept as a vector of feature weights. Features that exist in each movie are collected from the IMDb database (The Internet Movie Database, http://www.imdb.com). Feature weights are calculated prior to the start of the recommendation process, and weights are assigned to the features according to the degree of how well they can discriminate one movie from another, considering the characteristics of the whole movie domain. Therefore, feature weights are not kept in binary as in the approach described in (Berkovsky, Kuflik, and Ricci 2006). These feature weights and collaboratively built user models are used to find a weighted list of features that are liked or disliked by the user. Therefore the user preferences, interests and needs are modeled, using only the collaborative-based user model and item profiles. Users are enabled to view their proposed models. This enables them to be aware of their implicit or unknown preferences.

The proposed mechanism was implemented, and its accuracy was evaluated using the MovieLens million rating dataset (*http://www.grouplens.org*), which is a publicly available movie ratings dataset. As mentioned before, the IMDb database was exploited for extracting features of the rated movies for the casting, genre, year, director, writer, runtime, country, language, and color dimensions. The collaborative ratings provided in the dataset and the movie profiles formed after calculating each feature's weight in the whole domain, are used to construct a weighted list of liked and disliked features that will be used to test the proposed approach. The construction of user models in this manner was based on the assumption that a user's rating of a movie steadily

reflects his preferences concerning the features of the movies. Finally, the generated user models served as a basis for generating content-based recommendations. In this first phase of the evaluation process, we have completed two experiments. The first one was carried out to fine-tune the prediction generation and to find the best values for some threshold degrees kept in the system. The second one actually evaluated the accuracy of the predictions through the well known accuracy metrics, precision and recall (Herlocker et al. 1999). The second phase of the evaluation process is done by 30 active users, who evaluate the system's open user profile facility. Experimental results demonstrate the usefulness of the constructed content-based user models and the interest of the users towards the open user profile facility.

The remainder of the paper is organized as follows. First, we overview the related works, then we present the overview of our proposed approach in detail. Details of the experimental evaluation process are described next, and its results are discussed. Finally, the concluding remarks are offered and topics for further research are mentioned.

Related Work

When the movie domain is considered, the major approaches reported so far use collaborative filtering (Herlocker, Konstan, and Riedl 2000) and content-based filtering techniques (Tang, Winoto, and Chan 2003) (Melville, Mooney, and Nagarajan 2002) (Schein et al. 2002). A feature-weighted user-model generation is described in (Symeonidis et al. 2007). The well-known film recommender MovieLens (*http://movielens.umn.edu*) is provided by the GroupLens research project. It is based on collaborative filtering, which requires a sufficiently large number of ratings in order to achieve an appropriate recommendation.

There are few examples of open user profiles (Schafer, Konstan, and Reidl 2002) and almost no reported studies of open profiles that combine automated profile construction mechanism with transparency for the movies domain. MetaLens, a movie recommender system where user feedback is taken for different dimensions, is investigated in (Schafer, Konstan, and Reidl 2002). They take values for user preferences for different dimensions, but they do not enhance the user models implicitly using domain knowledge.

The incremental-critiquing approach at (McCarthy et al. 2005) describes a system where the user has the option of directly updating the candidates' selection query. However, the system builds the implicit user models incrementally through taking user feedback, and it does not include an automated user profile construction mechanism through item similarities, which is a desirable feature for end-user satisfaction.

In recommender systems, many works related to hybrid recommendation techniques (Balabanovic and Shoham 1997) (Jin, Zhou, and Mobasher 2005) (Salter and Antonopoulos 2006) tried to integrate multiple approaches in the prediction generation process (Burke 2002). MovieMagician (Grant and McCalla 2001) is a hybrid system that provides a rating prediction when requested. The features of a movie (kind, actors, and directors) are captured in a generic granularity hierarchy that is independent of a particular film and any specific movie is an instantiation of this hierarchy and the degree to which the instantiation hierarchies of two movies overlap defines their similarity. As a result, the features of a film can be used to find cliques, filter out irrelevant movies, annotate preferences about various features and generate explanations for a movie.

Hybrid recommenders usually combine two or more recommendation techniques, but they are not concerned with the conversion of user models between different techniques. In (Basu, Hirsh, and Cohen 1998), the authors extract content-based user models from collaborative user models and use both of these models for generating predictions. However, our approach focuses on the generation of pure content-based predictions, based solely on user models that are converted from collaborative user models with the efficient usage of domain knowledge. A content based user model generation is proposed in (Berkovsky, Kuflik, and Ricci 2006), which converts collaborative user models to content-based user models. However, they kept movie items as a set of features where all the features' weights are the same: 0 if a feature does not exist, 1 otherwise (in item profiles). In addition, we have used three main fine-tuning mechanisms, which results in higher precision than that of standard content based user models where movie item profiles are kept in binary.

System Description

Collaborative filtering is one of the most popular recommendation techniques which use cross-user correlations to generate predictions by weighing the opinions of similar users (Herlocker et al. 1999). The input to a standard collaborative filtering system is a matrix of users' ratings on a set of items, where each row represents the ratings of a single user and each column represents ratings on a single item. Thus, collaborative filtering user models are represented as a vector of pair of ratings i_k:r_k, which corresponds to a real rating r_k provided by the user on an item i_k.

Content-based filtering (Morita and Shinoda 1994) builds personalized recommendations by taking the features of items that have been rated by the user, and the set C of available items, not yet rated by the user, i.e., as input. The output of the system will be a subset of C, containing the items whose features match the features of the items that are liked by the user. Content-based recommenders generate predictions based on the set of features weighed according to a predefined scale. Therefore, the resulting user models can be represented as a vector of the pair of ratings f_k:w_k, where f_k denotes one of

the domain features and w_k *is* the level of the user's preference regarding this feature.

In addition to the user profiles, we keep item profiles in the system as a vector of pairs of f_i:w_i, where f_i denotes one of the features in a movie and w_i denotes its corresponding weight. The weight w_i is equal for all the features f_i, regardless of individual movies.

In the following subsections, we first describe the idea behind constructing item profiles and their construction algorithm. Then the user model construction algorithm is described. Finally, the recommendation generation step is discussed in detail.

Item Profile Construction

We consider each movie as a combination of dimensions where each dimension has a set of features. We base our item profile construction algorithm on the idea that the importance of each of the features in the domain should not be the same when we consider the degree of how well one discriminates a movie from the others; more specifically, (1) the ratio of the number the movies that has feature f_i to the total number of the movies in the database, (2) the number of the possible features in each dimension that feature f_i belongs to.

We propose that a feature f_i will be more discriminative if fewer movies in the whole movie domain have it. For instance, consider the following scenario: we have 10 ratings for a user, where in 5 of them actor *a1* acts, and the remaining 5 have country *c2*. Moreover, the user ratings for these movies are all the same. In addition, we have a total of 3500 movies in the domain and in 20 of them actor *a1* acted. Furthermore, there are a total of 600 films that have country *c2*. In this scenario, *a1* gives more clues about the user model, although both *a1* and *c2* exist in the same number of movies that has same ratings provided by the user. Although *a1* is not a very common feature in the domain, our user has rated movies in which *a1* exists. However, nearly 17% of all movies have the feature *c2* and it is more probable that the user has no consciousness of *c2* since it is a very common feature in the whole movie domain.

Our second hypothesis for the item profile construction process is that a feature will be more discriminative if the size of its dimension set is large. For instance, almost all movies have values from the genre and casting dimensions (except, possibly, animation movies). Therefore every movie has a set of features from each of these dimensions. If the size of the dimension set of a feature f_i is larger than another's, then the probability that the latter exists in any movie is smaller, which makes it a more descriptive feature. Consider again, for instance, the casting and genre dimensions. In our current database that we built for evaluation, there are 23 genres and 8409 actors/actresses in total. If an actor/actress acted in most of the movies that a user rated highly, this is more valuable information for us than knowing that a genre exists in most of the movies that the user rated highly.

In order to model our hypotheses, we used the Inverse Document Frequency (Sebastiani 2002) theory. We calculate the item feature weight (IFW) of each feature f_i with the following formula:

$$IFW\ (f_i) = \log\left(\frac{|M|}{|M_{f_i}|}\right) \times \log\left(|D_j|\right)$$

Here, $|M|$ is the size of the set of all movies in the domain, $|M_{fi}|$ is the total number of movies that has feature f_i,and $|D_j|$ is the size of the dimension that f_i belongs to.

In order to create user models, we use the collaborative user models and domain specific knowledge for the movies that we gather from the IMDb movie database. IMDB provided information in several dimensions; however for the sake of simplicity, we use 10 feature categories which we believe to be the most important ones for user preferences in our work: *genre, casting, language, year, country, rating, director, writer, runtime and color*. After this data is collected, we calculate *IFW* values for all the features in the domain and store each movie as pairs of f_i:w_i for each of the features f_i in each movie, where *wi* is the *IFW* value of f_i.

Construction of Content-Based User Models

We form content-based user models by using the collaborative user models and item profiles. We keep three weights for each feature f_i: *neg_weight_fi, pos_weight_fi, total_weight_fi. neg_weight_fi* corresponds to the weight of f_i that is collected from negatively rated movies, and *pos_weight_fi* corresponds to the weight of f_i collected from positively rated movies. (The reason behind keeping these weights separate is described in the Rating Estimation section). The total weight of a feature in the user model is kept in *total_weight_fi*, which is the sum of *neg_weight_fi* and *pos_weight_fi*.

For each movie rating provided in a collaborative user model, we have a list of a movie's features and their corresponding *IFW* values. We form the weights of each f_i in the user model by using the ratings of the movie that has f_i in its profile and its corresponding *IFW* value. Both the ratings used in OpenMore and the MovieLens dataset use a [1-5] scale. Ratings greater than 3 are taken as positive, and less than 4 as negative in almost all of the studies reported so far that use the MovieLens dataset. In order to highlight the negativity of the scores, we subtract 3 from each of the ratings. However, to stress the small negativity in rating 3, we do not take rating 3 as 3-3=0; we give -0.1 to highlight this effect and differentiate the features that occurred only in the movies that were rated with rating 3 from the ones that do not exist in the user model (does not exist in any of the rated movies).

The weights of the features in the user model are updated according to the rating of the movie, and the IFW values. In other words, the result of the multiplication of the rating of the movie with the corresponding *IFW* value

was added to the positive weights of all the movie genres, actors and directors involved in the movie, and similarly for all the remaining dimensions, if the rating for that movie is greater than 3. If the rating is smaller than 4, then negative weights of the features are updated in the same manner. In addition to this, the numbers of occurrences for each feature in negatively and positively rated movies having that feature are stored in the system (which will be used in the Rating Estimation).

For example, consider the rating *"The Usual Suspects":4* in the collaborative user model of one user. According to the IMDb, *"Stephan Baldwin" and "Kevin Spacey"* had roles in this movie. The user rated this movie positively; therefore the *pos_weight_fi* and *total_weight_fi* of these features are increased by the result of the multiplication of 1(4-3) with the *IFW* values of these features. The weights of other features that exist in this movie are updated accordingly. In addition to this, the number of occurrences in positively rated movies for these features is increased by 1.

We conclude the ideas behind the generation of a content-based user model below:

1. User models are kept as a vector of f_i*:(neg_weight_*f_i*, pos_weight_*f_i*, total_weight_*f_i*)* values, where each f_i exists in one of the rated movies of the user.
2. *neg_weight_*f_i for f_i is calculated by the following formula, where *c_neg_*f_i is the number of occurrences of f_i in negatively rated movies:

$$neg\_weight\_fi = IFW(f_i) \times c\_neg\_f_i$$

3. *pos_weight_*f_i for f_i is calculated by the following formula, where *c_pos_*f_i is the number of occurrences of f_i in positively rated movies:

$$pos\_weight\_fi = IFW(f_i) \times c\_pos\_f_i$$

4. *total_weight_* f_i for f_i is the sum of *neg_weight_*f_i *and pos_weight_* f_i.
5. For each f_i, we keep the number of occurrences of f_i in positively rated movies and negatively rated movies.

Optimizing Content-Based User Models

Considering the working domain, we use the following mechanisms in order to fine-tune the constructed content-based user models:

"Don't Care" Features

Content-based user models typically store features to which the user is indifferent. For instance, consider the following scenario: we have a user who sees only American movies, that is, he never prefers movies produced in any other country. Almost all of the movies he has rated have feature "*American*" for the country dimension. This shows us that the feature "*American*" has no effect on the preferences of the movies for that user, and the reason behind liking/not liking these movies that have the "*American*" feature are the other features except this feature; therefore this feature is identified as a "*don't care*" feature for the target user.

"Don't care" features are identified by the number of occurrences of that feature in the rated movies and the total number of movies that the user has rated. Identifying these features and removing them from the prediction generation step increases the accuracy of the results, because the recommender will base its decisions only on the differentiating features.

To filter these "*don't care*" features, a threshold is defined: "*TH_DONT_CARE*". Features that have the value *((c_pos_*f_i*+ c_neg_*f_i*)/total_rated_movies)* greater than "*TH_DONT_CARE*" are set to "*don't care*" and the prediction mechanism is designed and implemented such that only those features that are not "*don't care*" are used in the generation of the recommendation scores.

"Highly Positive" Features

When the movie domain is considered, there can be some features that the user prefers, and the user likes (has rated with high scores) all the movies that have that feature, regardless of other properties of those movies. For instance, consider a user who is a fan of "*Stephen Spielberg*" and likes all the movies directed by "*Stephen Spielberg*", regardless of other features in those movies. Therefore, if there is a movie directed by *"Stephen Spielberg*" that the user has not yet seen, there is a high probability that the user will like that movie, regardless of the features in casting, genre or other dimensions. As a result of this, we can identify these features as "*highly-positive*" features and promote them in the prediction score generation. After such features are identified, the prediction generation step uses the *pos_weight*s of the features that exist in any movie together with a "*highly positive*" feature, in order to be sure to come up with a positive score for such movies. However, if there are two or more movies with a "*highly positive*" feature, the one with the highest cumulated *pos_weights* of its features beats the others.

To identify "*highly positive*" features, two thresholds are defined: "*TH_HIGH_RATIO*", "*TH_HIGH_TOP_COUNT*". First the candidate's "*highly positive*" features are identified, which are the ones that have a *(c_pos_*f_i*/(c_pos_*f_i*+ c_neg_*f_i*))* value greater than "*TH_HIGH_RATIO*". Then these features are sorted according to their *pos_weights* and the top "*TH_HIGH_TOP_COUNT*" candidate's "*highly positive*" features that have the highest *pos_weights* are set to "*highly positive*". Therefore, we first eliminate the features that can not be "*highly positive*" by using only their number of occurrences in the positively and negatively rated films. Then we choose only those with the highest *pos_weights.* Therefore, we use the ratings provided for the movies that have those candidate "*highly positive*" features and their *IFW* values, in order to make second elimination.

"Highly Negative" Features

The idea behind "*highly negative*" features is the same as the idea behind "*highly positive*" features. With "*highly negative*" features, we try to identify which features always result in bad scores and use this valuable knowledge to fine-tune the constructed user models.

For instance, a user may dislike "*horror*" movies and never likes them regardless of other features that exist in a candidate movie that has the genre "*horror*". We identify "*highly negative*" features by using the same thresholds with the "*highly positive*" case. However, for the "*highly negative*" case, first elimination takes only the features that have a $(c\_neg\_f_i/(c\_pos\_f_i + c\_neg\_f_i))$ value greater than "*TH_HIGH_RATIO*". Then these features are sorted according to their *neg_weights* in order to take only the top "*TH_HIGH_TOP_COUNT*" ones that have highest *neg_weights* to "*highly negative*".

Further Details about Optimization of User Models

During optimization, we first identify the "*don't care*" features. Then the "*highly positive*" and the "*highly negative*" features are identified from the ones that have not already been set to "*don't care*". Therefore, a feature can never be set to both "*don't care*" and "*highly positive*" or "*highly negative*".

One more optimization is done on the remaining features that were not set to any of these values: if a feature f_i exists in negatively rated movies, and all those negatively rated movies have a "*highly negative*" feature, we set the *neg_weight* of that feature to 0, and we increase the *total_weight* accordingly (*total_weight* is set to *pos_weight*). The idea behind this optimization is that, the user did not like those negatively rated movies because of the "*highly negative*" features, not because of feature f_i. If this optimization results in features that have a *total_weight* equal to 0, then these features are removed from the user-model.

The same case does not hold for the features that exist in positively rated movies where all movies have a "*highly-positive*" feature. This was tested during the evaluation phase and resulted in a decrease in precision. So no optimization is done for this case.

Prediction Generation

The prediction generation process takes a content-based user model *um* and a set of candidate movies *MC*. It generates a list of recommended movies that are sorted according to their calculated recommendation score.

The prediction generation process produces scores for three distinct subsets of *MC* where each one is ordered in itself according to the produced score:

1. *List_Pos:* For the candidate movies in *MC* which have a feature that exists in *um* and is "*highly positive*".
2. *List_N:* For the candidate movies in *MC* which have no feature that exists in *um* and is "*highly positive*" or "*highly negative*".
3. *List_Neg:* For the candidate movies in *MC* which have a feature that exists in *um* and is "*highly negative*".

The resulting recommendation list has the candidates in *List_Pos* at the top. Then the candidates in *List_N*, and finally, the candidates in *List_Neg* exist in the recommendation list. The pseudo-codes for the generation of these lists are given below:

```
Gen_List_Pos(User-Mod um, candidate_movie_set MC )
  list_pos={}
  rec_score=0
  high_feature_exists=false
  for each mc∈MC
  {
    fSet(mc)=get feature set of mc
    for each f ∈fSet(mc)
    {
      if ( f ∈fSet(um) and is_dont_care(f,um)=false)
      {
        if(is_highly-pos(f,um)=true)
          high_feature_exists=true
        rec_score=rec_score+pos_weight(f)
      }
    }
    if (high_feature_exists=true)
      add mc to list_pos together with rec_score
    rec_score=0 }

Gen_List_N(User-Mod um, candidate_movie_set MC )
  list_N={}
  rec_score=0
  high_feature_exists=false
  for each mc∈MC
  {
    fSet(mc)=get feature set of mc
    for each f ∈fSet(mc)
    {
      if (   f ∈fSet(um) and
             is_dont_care(f,um)=false and
             is_highly-neg(f,um)=false and
             is_highly-pos(f,um)=false
         )
       {
         rec_score=rec_score+total_weight(f)
       }
      else
      {
        high_feature_exists=true
        exit for LOOP
      }
    }
    if (high_feature_exists=false)
      add mc to list_N together with rec_score
    rec_score=0 }
```

```
Gen_List_Neg(User-Mod um, candidate_movie_set MC )
  list_neg={}
  rec_score=0
  high_feature_exists=false
  for each mc∈MC
  {
    fSet(mc)=get feature set of mc
    for each f ∈fSet(mc)
    {
      if ( f ∈fSet(um) and is_dont_care(f,um)=false)
      {
        if(is_highly-neg(f,um)=true)
          high_feature_exists=true
        rec_score=rec_score+neg_weight(f)
      }
    }
    if (high_feature_exists=true)
      add mc to list_neg together with rec_score
    rec_score=0 }
```

Current Implementation

A content-based movie recommender, OpenMore is developed in order to evaluate the ideas presented in this paper. Since the proposed approach is tested using the MovieLens 1 million rating dataset, OpenMore retrieves all the content information from the IMDb database for the movies that exist in the dataset. As described in the Evaluation section, we performed evaluation in two phases: (1) an automated evaluation of the proposed approach using the collaborative user models formed from the MovieLens dataset is performed in order to determine the best values for the thresholds, and (2) a user study is performed (after the first phase of the evaluation is completed) using the same movies.

The users' interaction with OpenMore starts after the login phase. Users login to the system using their username and password. They have to rate at least 15 movies in order to start taking recommendations from the system. Users evaluate films that they have seen on a 5-point scale (5: masterpiece to 1: bad film). When they have rated (at least) 15 movies, user models are created in the system. Based on these models, a recommendation list is created as described in the previous section. A list of recommendations, sorted according to their prediction scores, is presented to the user. The movie with the highest prediction score is at the top of the list. Movies with positive prediction scores and movies with negative prediction scores are both presented to the user, together with their scores (which are normalized so that users view scores between -100 and 100).

Constructed user models are presented to the users in 2 main screens:

1. *Dimension Screen:* Dimensions are listed according to the total weight of the features in the user model that are summed up for each dimension. For instance, for the genre dimension, *total_weights* of all the genre features in the user model are summed up, and a genre dimension is assigned to that score. Every dimension is then listed in the order of this score in the *Dimension Screen*.
2. *Feature Screen*: When a dimension name in the *Dimension Screen* is clicked, the following sets of features are listed for the sets of features belonging to that chosen dimension.
   a. "*don't care*" *system_set:* features set by the system to "*don't care*"
   b. "*highly positive*" *set:* features set by the system to "*highly positive*".
   c. "*highly negative*" *set:* features set by the system to "*highly negative*".
   d. 10% of the features that exist in the user model with the highest *total_weight* values. These are the features that do not belong to the system constructed sets mentioned above.

Examples of the *Dimension Screen* and *Feature Screen (*for the language dimension*)* are displayed in Figure 1 and Figure 2 respectively.

Users can observe their constructed profiles and become conscious of their possibly unknown preferences for certain features of the movies through the *Dimension Screen* and the *Feature Screen*. This can also give them a reasoning mechanism for the predicted ratings, since the most important and effective features are listed in their user models.

Experimental Evaluation

In this section, we describe the experimental methodology and metrics we used to test our approach; and present the results of our experiments.

Data Set

Our experiments use the MovieLens million rating dataset, which is a collaborative filtering dataset storing ratings by 6040 users of 3952 movies. Ratings are provided on a scale of 1 to 5, 5 being excellent and 1 being terrible. Based on the rating guidelines presented to the users of MovieLens, we identified ratings of 4 and 5 to signify "good" movies (McLaughlin and Herlocker 2004). These are the movies that would make good recommendations.

As mentioned in the readme file of the dataset, a number of movieIDs do not correspond to any movie, due to some accidental duplicate entries and inconsistencies. As we are doing content-based filtering, we processed all the movies used in the dataset (in order to collect information from the IMDb database) and the inconsistent movie entries and their corresponding rating data are removed from the OpenMore database. Our resulting database has 3881 movie items, 6040 users and 1000187 ratings.

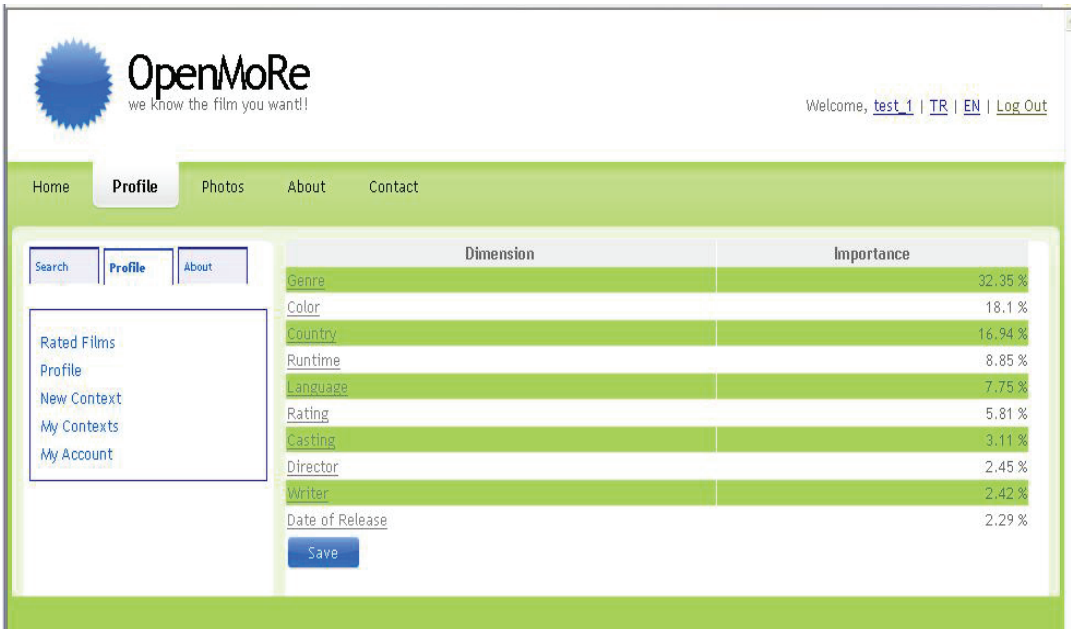

Figure 1: Dimension Screen

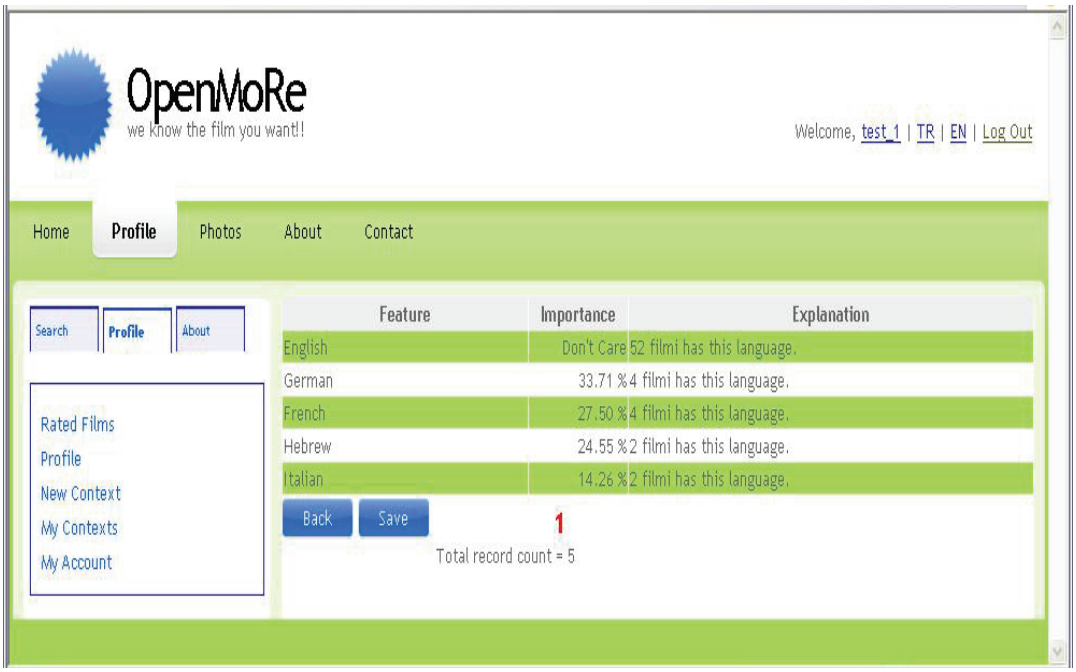

Figure 2: Feature Screen (for language dimension)

We applied 5-fold cross-validation to the dataset of the ratings by splitting the set of ratings using a 20%-80% ratio and doing this split 5 times for each user. For each of the 5 splits, we designate a 20% part of the initial dataset the *evaluation dataset* and the remaining 80% of the dataset was designated the *training dataset*. We repeat our experiments with every training set and test set for each of the users selected for evaluation, and after this phase is completed we average their results. We generate each testing set by taking a random sample (Basu, Hirsh, and Cohen 1998) of the data as follows:

- For every user, we separate and group his movie/rating pairs into intervals defined by ratings.
- From each interval, we take the same number of movie/rating pairs (when possible, since sometimes the number of items is not divided evenly by 5), and adjust the distribution in order to have a 20% distribution of movie/rating pairs each time.

As a result of this procedure, each of the testing sets is more representative of the distribution of ratings for the entire data set than would have been the case with simple random sampling.

In our experiments, we select *2872* users whose variance of ratings is not *0* (i.e., the ratings are not identical). Table 1 shows the distribution of the number of rated movies among the users in the dataset.

Evaluation Metrics

There are several performance metrics that are commonly used to evaluate the performance of recommender systems, such as the mean absolute error (MAE), mean squared error (MSE), precision, recall, and F-measure (Mooney 1999) (Herlocker et al. 1999). Moreover, (Herlocker et al. 1999) classifies these metrics into *statistical accuracy* and *decision-support accuracy metrics.* The statistical accuracy metrics compare the predicted ratings against the actual user ratings on the test data. The MAE measure is a representative example of a statistical accuracy measure. The decision-support accuracy metrics, like precision, recall, and F-measure, measure how well a recommender system can predict which of a set of unknown items will be highly rated.

Moreover, although both types of measures are important, it has been argued in the literature (Herlocker et al. 1999) that decision-support metrics are better suited for recommender systems, because they focus on recommending high-quality items, which is the primary target of recommender systems. Therefore we used precision and recall metrics in our evaluation. Precision is defined as the fraction of positive examples classified as positive that are actually positive. Recall is defined as the fraction of positive examples classified as positive.

number of rated movies	number of users
0 to 25	492
26 to 50	1301
51 to 75	785
76 to 100	553
101 to 125	480
126 to 150	345
151 to 175	306
176 to 200	200
201 to 225	207
226 to 250	148
251 to 300	268
301 to 500	559
over 500	396

Table 1: Distribution of ratings among the users in the dataset

Evaluation Phases

We completed the evaluation process in two phases:

(1) *Automated Evaluation Using MovieLens Dataset:* The first phase of the experiment is accomplished using the MovieLens collaborative filtering dataset. The collaborative user models in the dataset are transformed to content-based user models and two sets of experiments are performed.

First Set of Experiments

The first set of experiments was designed to fine-tune the prediction mechanism by selecting the most appropriate values for the *TH_DONT_CARE*, *TH_HIGH_TOP_COUNT* and *TH_HIGH_RATIO* thresholds. To accomplish this, we set two of the thresholds to a constant, while the values of the remaining ones are gradually modified. For each value of the modified threshold, a subset of *500* users that rated at least *100* movies is selected, and for each user, we performed a 5-fold cross-validation on his rating data. In this way we completed 5 test runs for each user, which makes a total of 2500 test runs. Results of the predictions are evaluated using the metrics mentioned in the Evaluation Metrics section.

1. Evaluation for *TH_DONT_CARE*

To find the most appropriate value of *TH_DONT_CARE, TH_HIGH_TOP_COUNT* and *TH_HIGH_RATIO* thresholds are set to *0.80*. The values of *TH_DONT_CARE* are increased from *0.70* to *0.95* by *0.5*. Table 2 illustrates the results of the experiments.

TH_DONT_CARE	Precision	Recall
0.70	64.6%	59.2%
0.75	66.0%	59.4%
0.80	72.2%	66.6%
0.85	73.1%	72.8%
0.90	69.6%	78.7%
0.95	67.2%	80.6%

Table 2: Results of evaluation for TH_DONT_CARE

As can be seen, precision increases with increased *TH_DONT_CARE* value up to 0.85 and decreases afterwards. The increase can be explained by the identification of more "*don't care*" features incorrectly with small values of *TH_DONT_CARE*. However, with values higher than 0.85, precision decreases, which will be the influence of missing some "*don't care*" features because of high *TH_DONT_CARE.* As expected, recall behaves in a similar manner; however, its value increases up to *0.90*. By taking a weighted average of precision and recall, *TH_DONT_CARE = 0.85* is taken as an optimal value.

2. Evaluation for *TH_HIGH_TOP_COUNT*

After determining the value for *TH_DONT_CARE*, it is used to choose an optimal value for *TH_HIGH_TOP_COUNT.* The *TH_HIGH_RATIO* remains as *0.85*. Similar to the previous example, the values for *TH_HIGH_TOP_COUNT* are increased from *0.70* to *0.95* by *0.5*. Table 3 illustrates the results of the experiments.

TH_HIGH_TOP_COUNT	Precision	Recall
0.70	59.7%	66.4%
0.75	69.2%	69.3%
0.80	72.0%	71.5%
0.85	73.7%	72.3%
0.90	74.1%	73.7%
0.95	70.2%	73.9%

Table 3: Results of evaluation for TH_HIGH_TOP_COUNT

With arguments similar to those used in the first experiment, low values of *TH_HIGH_TOP_COUNT* lead to incorrect analysis of "*highly positive*" features, which decreases the precision values. The best results are taken when *TH_HIGH_TOP_COUNT is* set to *0.90*. Therefore we take 0.90 as an optimal value for *TH_HIGH_TOP_COUNT.*

3. Evaluation for *TH_HIGH_RATIO*

After determining the value for *TH_DONT_CARE*, *TH_HIGH_TOP_COUNT,* we conducted experiments for *TH_HIGH_RATIO* by setting the other thresholds to their observed optimal values. The values for *TH_HIGH_RATIO* were increased from *0.78* to *0.90* by *0.2*. As can be observed from the results in Table 4, we achieved the best results with 0.88.

TH_HIGH_RATIO	Precision	Recall
0.76	61.2%	63.2%
0.78	64.2%	63.8%
0.80	72.6%	65.1%
0.82	72.9%	66.0%
0.84	73.8%	68.2%
0.86	74.0%	72.1%
0.88	74.5%	73.2%
0.90	72.5%	72.7%

Table 4: Results of evaluation for TH_HIGH_RATIO

Second Set of Experiments

The determined threshold values are applied in the second set of experiments. In this step, we try to figure out the influence of four optimizations over a purely content-based user model generation. We performed 5 tests:

1. *Test 1:* evaluation of a purely content-based user-model generation without any optimization.
2. Test 2: evaluation of the influence of "*don't care*" features with a purely content-based user model generation
3. Test 3: evaluation of the influence of "*highly*" features (both "*highly positive*" and "*highly negative*") with a purely content-based user model generation
4. Test 4: evaluation of the influence of "*highly*" features (both "*highly positive*" and "*highly negative*") with a purely content-based user model generation and the last hypothesized optimization that is, updating the *neg_weights* of features that appear in negatively rated movies that all have a "*highly negative*" feature.
5. Test 5: evaluation of content-based user-model generation with all the optimizations.

For these experiments, *2872* users (selection described in Data Set section) are selected and used with 5-fold cross-validation on the rating data. The results are summarized in Table 5.

TEST NUMBER	Precision	Recall
1	70.2%	89.1%
2	72.9%	73.9%
3	71.9%	84.3%
4	72.3%	84.1%
5	75.2%	73.7%

Table 5: Results of evaluation of the system for determining the influences of different optimizations

As can be seen from the results, identification of "*highly*" features greatly increased precision when compared to the identification of "*don't care*" features. Recall decreased with "*don't care*" feature identification, which is expected, since we eliminate a set of features from the user-model which results in producing predictions for a smaller number of movies. Our results indicate that we achieve the highest precision when we integrate all the proposed optimizations into the system, although recall decreases, which is an expected result.

In (Christakou and Stafylopatis 2005), the performance results for different recommenders are given in terms of precision and recall metrics; however, information regarding the dataset used in the evaluation and the evaluation procedures were not mentioned exactly. The results from (Christakou and Stafylopatis 2005) are displayed in Table 6 in order to compare the systems' performance with the existing studies. MovieLens, is a collaborative filtering recommender and MovieMagician (Grant and McCalla 2001) is a hybrid recommender system that provides a rating prediction when requested.

Methodology	Precision (%)	Recall (%)
MovieLens	66	74
MovieMagician Feature-Based	61	75
MovieMagician Clique-Based	74	73
MovieMagician Hybrid	73	56
OPENMORE	75.2%	73.7%

Table 6: Comparative Performance Results

As it can be seen from the results, our system's precision values are higher than MovieLens' and MovieMagician'. There is a small difference in recall when MovieLens and Feature-Based MovieMagician is considered, which can be the effect of the "don't care" features. For instance, the system cannot make predictions for the movies that have only "*don't care*" features common with the constructed user models, which results in a decrease in recall.

The main advantage of our system over collaborative systems is that, our system can generate a recommendation score for every movie that has common feature(s) with the constructed user models that are not "don't care"s. However, most of the collaborative-based methods cannot produce predictions for the movies that were not rated by any peer users.

(2) *Evaluation of Open User Models:* A user study was performed with 30 students who were willing to use a movie recommendation system. This phase of the evaluation aimed to observe the influence of the open user models on the end user satisfaction.

Users were first informed about the system. They were told how they should proceed before taking recommendations, and how they could view their profiles. The experiment took 15 days, and the users were asked to complete a questionnaire after the evaluation phase of OpenMore was completed. The questions were framed in a 5 point Likert scale and involved the following statements:

1. Generally speaking, I like the movie that the system recommends.
2. I find the recommendations useful.
3. It is easier to decide on which movie to watch with OpenMore.
4. The viewing profile is very useful.
5. The recommender makes the decision making process more efficient through open user models.
6. It is fun to use OpenMoRe.
7. All in all, I would like to use OpenMoRe when I need to find which movie to watch.

The results of the questionnaire are displayed in Table 7. They show that the reactions of the subjects were generally supportive regarding proposed recommendations, and open user profiles. From Table 7 we can see that more than 80% of the subjects agreed or strongly agreed that the system is useful, and makes the decision-making process easier. The open user profile feature was mentioned as useful by more than 85% of the subjects. Some of the users' mentioned that it would be better to view their profiles on more user-friendly screens. In addition, they wanted to see why some features were selected as "*don't care*", "*highly positive*" and "*highly negative*".

Question Number	Average Answer
1	4.1
2	4.7
3	4.2
4	4.8
5	4.8
6	3.9
7	3.8

Table 7: Results of the questionnaire

Even though these statistical results are from a pilot study, at least up to this stage the results encourage further work on open user models. If the user models presented in this paper are made more understandable and if some explanations are provided for the selected "*don't care*", "*highly positive*" and "*highly negative*" features, this may increase end user satisfaction.

Conclusions and Future Research

This paper presents a content-based approach to movie recommendation, with open user profiles. The system first constructs item profiles by using degrees of discriminativeness for features from the whole movie domain. It transforms the collaborative user models to content-based user models by using rating data provided and constructed item profiles. Recommendations are generated using the constructed user models.

The experimental study first focused on determining the thresholds that are used in the fine-tuning of content-based user models. The thresholds were then applied and the accuracy of the generated content-based predictions was evaluated. The experiments showed that optimization mechanisms on the built user models result in better recommendations. The next step of our study will be focusing on the effect of the whole dimension together with the effect of features only. We believe that this will significantly improve results.

References

Adomavicius, G., and Tuzhilin, A. 2005. Towards the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17 (6): 734–749.

Balabanovic, M., and Shoham, Y. 1997. Fab: Content-based, collaborative recommendation. *ACM Communications*, (3):66–72.

Basu, C., Hirsh, H., and Cohen, W. 1998. Recommendation as classification: Using social and content-based information in recommendation. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, 714–720.

Berkovsky, S., Kuflik, T., and Ricci, F. 2006. Cross-Technique Mediation of User Models. In *Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, (AH),* 21-30.

Burke, R. 2002. Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*, 12 (4): 331-370.

Christakou, C., and Stafylopatis, A. 2005. A Hybrid Movie Recommender System Based on Neural Networks. In *Proceedings of the 5th International Conference on Intelligent Systems Design and Applications*, 500-509.

Grant, S., and McCalla, G. I. 2001. A hybrid approach to making recommendations and its application to the movie domain. In *Proceedings of the Canadian AI Conference (AI 2001),* 257-266.

Herlocker, J. L., Konstan, J. A., Borchers, A., and Riedl, J. 1999. An Algorithmic Framework for Performing Collaborative Filtering. In *Proceedings of SIGIR Conference*, 230-237.

Herlocker, J., Konstan, J., and Riedl, J. 2000. Explaining Collaborative Filtering Recommendations. In *Proceedings of the ACM 2000 Conference on Computer Supported Cooperative Work*, 241-250.

Jin, X., Zhou, Y., and Mobasher, B. 2005. A Maximum Entropy Web Recommendation System: Combining Collaborative and Content Features. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'05).*

McCarthy, K., Reilly, J., McGinty, L., and Smyth, B. 2005. An Analysis of Critique Diversity in Case-Based Recommendation In *Proceedings of the Eighteenth International Florida Artificial Intelligence Reseach Society Conference (FLAIRS-05),* 123-128.

McLaughlin, M.R., and Herlocker, J.L. 2004. A collaborative filtering algorithm and evaluation metric that accurately model the user experience. In *Proceedings of the 27th annual international conference on Research and development in information retrieval (SIGIR 2004)*, 329-336.

Melville, P., Mooney, R. J., and Nagarajan, R. 2002. Content-Boosted Collaborative Filtering for Improved Recommendations. In *Proceedings of the AAAI-02*, 187-192.

Mooney, R. J. 1999. Content-based book recommending using learning for text categorization. In *Proceedings of the SIGIR-99 Workshop on Recommender Systems:Algorithms and Evaluation.*

Morita, M., and Shinoda, Y. 1994. Information Filtering Based on User Behavior Analysis and Best Match Text Retrieval. In *Proceedings of SIGIR Conference*, 272-281.

Schafer, J. B., Konstan, J. A., and Reidl, J. 2002. Meta-recommender Systems: User-controlled Integration of Diverse Recommendations. In *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM-02),* 196-204.

Salter, J., and Antonopoulos, N. 2006. Cinemascreen recommender agent: Combining collaborative and content-based filtering. *Intelligent Systems Magazine*, 21(1):35–4.

Schein, A. I., Popescul, A., Ungar, L. H., and Pennock, D. M. 2002. Methods and Metrics for Cold-Start Recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval,* 253-260.

Sebastiani, F. 2002. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1): 1-47.

Symeonidis, P., Nanopoulos, A., Papadopoulos, A.N., and Manolopoulos, Y. 2007. Feature-weighted User Model for Recommender Systems. In *Proceedings of the 11th International Conference on User Modelling (UM 2007).*

Tang, Y. T., Winoto, P., and Chan, K. C. C. 2003. On the Temporal Analysis for Improved Hybrid Recommendations. *IEEE/WIC International Conference on Web Intelligence*, 214-220.