Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2016) 24: 833 – 848

doi:10.3906/elk-1303-189

Feature selection for movie recommendation

Zehra ÇATALTEPE[1,*], **Mahiye ULUYAĞMUR**[1], **Esengül TAYFUR**[2]
[1]Department of Computer Engineering, İstanbul Technical University, Maslak, İstanbul, Turkey
[2]Digiturk, Beşiktaş, İstanbul, Turkey

Received: 26.03.2013 • **Accepted/Published Online:** 04.11.2013 • **Final Version:** 23.03.2016

Abstract: TV users have an abundance of different movies they could choose from, and with the quantity and quality of data available both on user behavior and content, better recommenders are possible. In this paper, we evaluate and combine different content-based and collaborative recommendation methods for a Turkish movie recommendation system. Our recommendation methods can make use of user behavior, different types of content features, and other users' behavior to predict movie ratings. We gather different types of data on movies, such as the description, actors, directors, year, and genre. We use natural language processing methods to convert the Turkish movie descriptions into keyword vectors. Then, for each user, we use the content features and the user's past implicit ratings to produce content feature-based user profiles. In order to have more reliable profiles, we do feature selection on these profiles. We show that for each feature space, such as actor, director, or keyword, a different amount of feature selection may be optimal. Different recommenders may also perform best for a different number of movies available as training data for a user. We also combine different content-based recommenders and collaborative recommenders using an aggregation or the best of the available recommenders. Experimental results on a dataset with hundreds of users and movies show that, especially for users who have watched a small number of movies in the past, feature selection can increase recommendation success.

Key words: Feature selection, content feature-based user profile, content-based filtering, hybrid recommenders, machine learning

1. Introduction

Over the last couple of decades, digital TV broadcasting content available for users has increased in many fields, while users' time has not. Finding and watching content that a user likes by zapping through the channels or the TV guide has become an impossible task. Therefore, selection of suitable content for each user has become an important problem [1,2]. Recommender systems have emerged as a solution to this problem.

There are many taxonomies of recommendation systems. If explicit ratings (e.g., like/dislike, a discrete rating) are available, then a recommendation system may use these ratings. On the other hand, for many systems, users do not want to provide such explicit feedback, and therefore implicit ratings need to be produced based on the user's past behavior. Another taxonomy of recommendation systems is based on whether the content of each movie or the viewing behavior of other users is taken into account. Usually, collaborative filtering methods ask users to give explicit ratings about the contents they watched previously, so that the ratings are obtained explicitly by the system [3,4]. If ratings are not available, then they may need to be generated implicitly, based on user behavior. Sparsity of the user-item matrix is a major problem in collaborative filtering [5]. There are usually a large number movies in the system and each user gives ratings to a small number of movies. Since

*Correspondence: cataltepe@itu.edu.tr

the number of commonly rated movies by two users is generally zero, it becomes very difficult to find users who are nearby. Moreover, if a movie is not rated by any users at all, it cannot be recommended (cold-start problem) [5]. On the other hand, content-based recommenders use movie information and users' behavior that is usually summarized in user profiles. The Music Genome Project is an example of a content-based music recommendation system [6]. In a content-based method, each user is uniquely characterized and the user's interest is not matched with some other user as in the collaborative methods [7]. The ability to show content features that cause an item to be recommended also gives users confidence about the recommendation system and insight into their own preferences [7]. Recommender systems have been evaluated based on the accuracy of the recommendations predicted; however, some studies [8,9] emphasized that user experience should be a part of this evaluation. We think that explaining recommendations based on the content features makes the recommender more transparent to the user. In addition to collaborative and content-based recommendation methods, there are many different hybrid recommendation methods [10].

Different solutions based on explicit or implicit ratings or collaborative, content-based or hybrid methods have been proposed for the TV recommendation problem [1,11,12]. The TV movie recommendation task to a user on a certain day has the following challenges, which may not be the case for most of the other recommendation tasks. First of all, explicit ratings are not available and implicit ratings need to be computed based on the users' past behaviors. Another issue is the fact that TV movies need to be recommended to a whole family as opposed to an individual, and therefore the implicit ratings reflect the likes/dislikes of not an individual but a number of them. There is also a dynamic set of movies available each day and therefore the recommender needs to be able to work on a different set of items every day. Finally, since the recommendation is usually shown on a TV screen and is for a single day, there needs to be a short list of up to 3 or 4 recommendations and therefore recommenders need to be evaluated for their performance regarding that small list.

Contributions of our paper can be summarized as follows:

- Ours is the first work on Turkish movie recommendation that compares different types of content (such as keywords, actors, directors, genre, year)-based recommendation and collaborative filtering.
- Our recommendation method does not rely on explicit ratings, which are hard to get; instead it uses implicit ratings computed based on user behavior.
- We evaluate different recommendation combination methods, namely hybrid, sum, and best combiner. These combination methods not only combine collaborative and content-based recommenders, but also different content-based recommenders.
- We provide a fast filter-type feature selection method that is used on content features. We show that feature selection may need to be used in different degrees for each feature space.
- We present experimental results on a dataset consisting of hundreds of users and movies.

The rest of the paper is organized as follows. In Section 2, we present the related work in the movie recommendation literature. In Section 3, we describe how implicit ratings and features are produced and how feature selection is performed. In Section 4, content-based, collaborative, and hybrid recommendation methods used are described. Evaluation methods are given in Section 5. Section 6 presents and discusses the experimental results. We conclude the paper with Section 7.

2. Related work

With the increase in the amount of items that users can buy/watch and the ability to keep the history of users' consumed items, recommendation systems have become very necessary. A recommendation method for a Japanese video service provider was proposed in [13]. It used the actor and keyword information of the users' watched films and also considered the time of the day the users watch TV. It used the ratio of the number of times a user watched a movie with a certain feature (such as actor, keyword) to the number of times the feature was observed in all the movies. This ratio is calculated for all actor and keyword features. Then, for each movie, the sum of the movie's ratio features is calculated. Recall, precision, and F-measure are used as evaluation measures. Different from our evaluation method, it measured performance by getting feedback from the user right after making a recommendation. An approach to support context-aware recommendation for personalized digital TV was proposed in [1]. In this work, the authors used a contextual user profile, which consists of the user's personal data profile, user contextual information, and the genre of the TV program. This information was obtained from the users by asking them directly.

The distinction of our system is that we do not ask any questions to the users to get their demographic information or preferences. As each user may correspond to a number of people in our system and the information provided may not always be reliable, we do not use the demographic information of users at all. The FIT system recommends TV programs to family members [11]. In that study, Goren-Bar and Glinansky constructed a user profile by asking each household about their program genre preferences. The user profile also contained the times of the day they watch television. In the recommendation phase, first the FIT system guesses which household turns on the television by using the time of the day information. If the guess is wrong, then the system may make the wrong recommendations for the household. The TiVo television recommendation system uses an item-item form of collaborative filtering [2]. The process starts by a user giving a rating to a movie. There are two types of ratings: explicit and implicit. For explicit feedback, the user presses the remote control button according to how much s/he loves the movie.

We use content feature-based user profiles that are related to the content-based profiles described in [14]. In that study, the authors used WordNet to find synonyms and, instead of bag of words, they used a bag of synsets to represent user profiles. They used slots of keywords where they keep movies' titles, casts, directors, and a summary and keywords. Our work is in Turkish and can be extended using Balkanet [15,16] data, which includes a set of synonyms for Turkish, but is not as extended as Wordnet. Our work differs from [14] in a number of ways. In our study, we do not have explicit ratings obtained from the users. We need to compute the implicit ratings based on the user behavior. We include these implicit user ratings in content feature-based user profiles, we do feature selection in each profile for each user separately, and we also combine different content-based ratings and collaborative ratings.

A TV recommender system for multiple viewers called TV4M was introduced in [12], where Yu et al. demonstrated their system with 25 users and 200 movies. Each user has a separate user profile. They represent each term in the user profiles based on explicit user feedback or using relevance feedback. They do feature selection on each user profile based on whether the feature contains dominating negative or positive feedback. Then they merge the user profiles to produce the group profile. In our work, we do not have the explicit ratings of the individual users and have to work with the implicit ratings for the whole household. We incorporate these continuous-valued implicit ratings and different types of content features, such as movie actor, director, keyword, genre, and year, in the user profiles. We do feature selection for each content feature type separately. We merge the ratings produced by each different content feature type and also collaborative ratings. Another

related work is [17]. In that paper, the authors used a sequential forward feature selection algorithm on movie features. In our work we use a fast filter type [18] of feature selection algorithm and also use the keyword features in addition to the other features. We do feature selection on each feature set separately and we also combine different collaborative and content-based ratings.

In another related study [19], we used implicit ratings and used actor, director, genre, and keyword features for movie recommendation. We also combined different predicted ratings using sum and weighted sum methods. In this study, we use feature selection on different feature sets and we also use a new recommender combination method.

3. Methodology

3.1. Implicit rating computation

If explicit ratings (e.g., like/dislike, a discrete rating) are available, then a recommendation system may use these ratings. On the other hand, for many systems, users do not want to provide such explicit feedback, and therefore implicit ratings need to be produced based on the user's past behavior. Most recommendation systems rely on explicit user feedback [3,4]. If ratings are not available, then they may need to be generated implicitly based on user behavior. It should be emphasized that using direct ratings or calculated implicit ratings may produce different results.

In our system, users do not rate movies explicitly, so we calculate implicit ratings by using the viewing durations. Assume that user u watches the movie i for $t(u,i)$ minutes during the training period and t_i is the total duration of the movie i. We define the normalized viewing duration (the implicit rating) of user u for movie i as:

$$r(u,i) = \frac{t(u,i)}{t_i}. \tag{1}$$

This formulation is similar to that of [20]. The difference in our formulation is that it can be greater than 1 if a user has watched a content more than once.

Furthermore, we work with continuous ratings; we do not use a threshold to determine relevant items. As seen in Figure 1, values are between 0 and 7.2. Most contents are watched once, but there are also contents that are watched more than once. Users who started watching a content and then left it from the peak are $r(u,i) = 0$ and a large portion of users finish the whole content (the peak at 1) instead of leaving it in the middle.

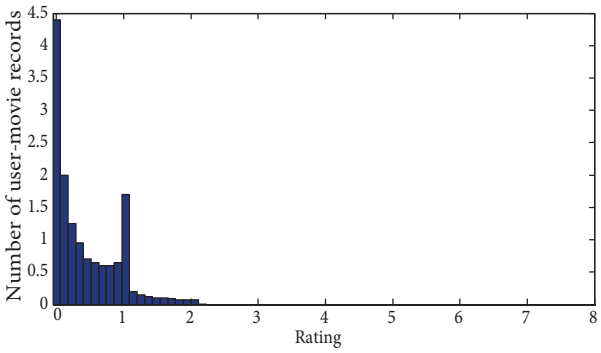

Figure 1. Distribution of the implicit ratings computed (the values on the vertical axis are multiples of 10,000).

3.2. Content features

The movie actor, director, and genre features were used directly.

The release year information was discretized into six different values as before 1980, 1980s, 1990s, 2000s, 2010–2011, and 2012.

The movie synopsis documents were used to get the keywords of a movie. First of all, the synopsis documents were processed using the Zemberek software [21]. Zemberek is an open-source, platform-independent natural language processing software for Turkish and other Turkic languages. In the Turkish language, new words are formed by concatenating suffixes on top of the root. Zemberek finds the root(s) that can be the actual root of an observed word. Zemberek cannot process non-Turkish words, such as Bahamas, New York, etc. Instead, these words can be used just the way they are for content feature representation. After we obtain the roots of the words in the summary document, we need to use them to represent a movie. In order to extract the keyword features for movies, similar to [14], we use the tf-idf weights [22]. The tf-idf weight keyword (term) j in movie i is computed as:

$$tf - idf\,(j,i) = tf\,(j,i) \times \log(N/n_j). \tag{2}$$

In this equation, term frequency, $tf\,(j,i)$ is the number of times keyword j occurs in movie i, and inverse document frequency is $\log(N/n_j)$ where N is the total number of movies considered and n_j is the number of movies that have the keyword j.

In Table 1, we give example content features for two different movies.

Table 1. Example content features for two movies.

Movie	Genre	Actor	Director	Keyword
The Shawshank Redemption	Crime Drama	Tim Robins Morgan Freeman Bob Gunton	Frank Darabont	hapishane, dost, cinayet, esaret (prison, friend, murder, slavery)
Muppets from Space	Family	Dave Goelz Bill Barretta	Tim Hill	gezegen, kukla, merak, sev, uzay (planet, puppet, curiosity, love, space)

3.3. Content feature-based user profiles

The aim of the movie recommendation system is to find the contents that the user may actually want to watch. We use the features of the movies the user viewed in the past and the implicit ratings of the users for these movies in order to compute a weight for each user and feature. We use the feature set types of actor, director, keyword, genre, and release year. Training contents have 5074 actors, 2159 directors, 5223 keywords, and 36 genres and have been categorized according to 6 release year values. The weight of each feature is assigned according to the implicit ratings of the user for all training data contents including that feature. The actor feature set may contain features like Brad Pitt, Harrison Ford, or Engin Günaydın; the genre feature set may contain drama, action, and comedy; the director feature set may contain Woody Allen, James Cameron, etc. In order to determine the weight of each feature for user u, we use the rating data of the training period.

Let user u watch movies $i_0, ..., i_8$, which have features $j_0, ..., j_4$. In Table 2, we show the features for the feature set k for user u. Note that $j_0, ..., j_4$ contain the features that are related to the movies all users watched within the training period. The rating column shows the rating of user u for movies $i_0, ..., i_8$.

Table 2. An example content feature-based profile computation.

user u	j_0	j_1	j_2	j_3	$j_4 \ldots .$	$r(u,i)$
i_0	1	1	0	0	0	0.5
i_1	0	1	0	0	0	0.3
i_2	1	1	1	0	0	0.9
i_3	1	0	0	1	0	0.7
i_4	0	0	0	1	0	0.2
i_5	1	0	0	1	0	1.0
i_6	1	0	0	1	0	0.44
i_7	0	1	0	0	0	0.67
i_8	1	0	0	1	0	0.2
$w_k(u,j)$	0.42	0.26	0.1	0.26	0	-

The weight of feature j in feature set k for user u is calculated as:

$$w_k(u,j) = \frac{1}{|I_u^{train}|} \times \sum_{i \in I_u^{train}} x_{k,u}(i,j) \times r(u,i). \tag{3}$$

In Eq. (3), I_u^{train} is the set of the movies watched by user u within the training period. k represents the type of the feature set, $k \in \{$actor, director, keyword, genre, release year$\}$. $r(u,i) \in R$ is the implicit rating of user u for movie i and $x_{k,u}(i,j) \in \{0,1\}$ is the jth feature of movie i. If movie i has feature j, then $x_{k,u}(i,j)$ will be 1 and the user rating for movie i will contribute to the sum.

Table 2 shows an example of content feature-based user profile computation.

In Figure 2, we show the keyword, actor, director, and genre content feature-based profiles for three different users. Each user may have watched a different number of movies within the training period; therefore, the user profiles contain different numbers of elements. The number of features in the profile reduces as the feature types change from keyword, to actor, to director, and then genre. It is also possible to deduce some information about the user's preference type based on the profiles. For example, among these three users, user 3 prefers some specific actors and directors, whereas user 2 prefers a more widespread range of actors and directors. In fact, we looked closer and found out that user 3 prefers Turkish classical movie actors, such as Türkan Şoray, Ediz Hun, and Zeki Alasya.

3.4. Feature selection

Feature selection methods [18] have long been used in machine learning to find the most relevant features for classification [23,24] or to get rid of noisy or redundant features. When redundant or noisy features are cleaned, simpler models, which can generalize better, can be trained and tested in shorter time. For the TV recommendation problem that we consider, we cannot use positive or negative ratings of all movies; instead, we can only use the implicit ratings of the movies the users have watched. Therefore, it is not feasible to use a correlation measure, such as mutual information [25] as in [23], to measure the relevance of each feature.

In order to compute the relevance or redundancy of movie features, we need to consider each user separately, because while a feature may be important for one user, it may not be important for the others. In the content feature-based user profile entry (Eq. (3)), $w_k(u, j)$ is proportional to how many movies containing the feature j user u has watched and how much (through the implicit rating) s/he liked them. Therefore, as a

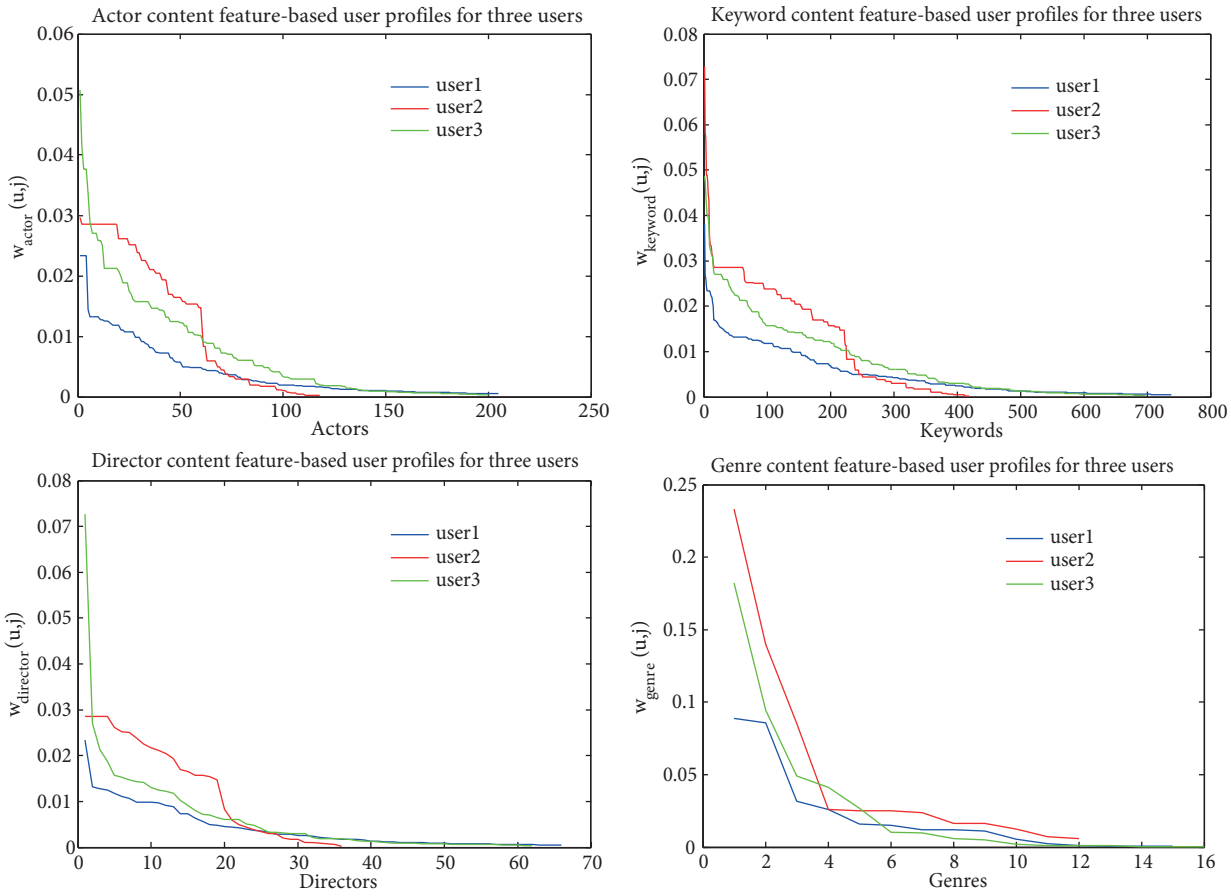

Figure 2. Keyword, actor, director, and genre content feature-based profiles for three different users.

measure of relevance of feature j for user u, we use the content feature based user profile entry:

$$Rel\,(u,j) = w_k(u,j). \tag{4}$$

In most classification problems, if two features are correlated, since both features are available, one can be omitted. We could define correlation between two features j and $j^{'}$ as how close their $w_k(u,j)$ and $w_k(u,j^{'})$ values are. However, for content-based recommender systems, if a movie contains even only some of the relevant keywords, those keywords should be kept in the user profile. Therefore, in order to select the features for movie recommendation, we do not get rid of the redundant features but we do disregard the irrelevant ones. Similar to [23,24], which ordered features according to their relevance values, we order features and select only features that have the highest $w_k(u,j)$ values. In the experiments, we experiment with using only the topmost p percent of the features and find out that using feature selection can improve the performance of the recommender. In order to select the best ratio of the features, we use the average test performance over all users.

4. Recommendation methods

4.1. Content-based recommendation

After calculating the weights of each feature for each user, we use them to predict the recommendation ratings of the contents. We calculate the rating for each feature set separately using the weights obtained with the use

of the training data. We use the following method to generate ratings:

$$r_k(u,i) = \sum_{j \in F_{k,j}} w_k(u,j). \tag{5}$$

In this equation, $r_k(u,i)$ represents the rating that user u gives movie i according to the feature set k. $F_{k,j}$ are the features under feature set k that are related to movie i.

4.2. Collaborative recommendation

The basic idea behind collaborative filtering is that when we need to predict the rating for a certain user and a certain movie, even if the user has not seen the movie, we can check to see if other people with whom the user had similar ratings on past movies (i.e. the user's neighbors) liked that movie or not and we can base the predictions on those neighbors' opinions.

We formalize this basic idea as follows: Consider that we need to predict the rating of user u for movie i. Let D_i be the set of users who have watched the movie i within the training period. If $D_i = \emptyset$, then we can not produce a collaborative rating. Let $D_{i,u} \subseteq D_i$ be the set of users who have also watched at least one common movie with u within the training period. Then we compute the collaborative rating of user u for movie i as:

$$r_c(u,i) = \frac{1}{C} \times \sum_{v \in D_{i,u}} c(u,v) \times r(v,i). \tag{6}$$

In this equation, $c(u,v) \geq 1$ is the number of movies that both user u and user v have watched within the training period. $r(v,i)$ is the implicit rating of user v for movie i. C is a normalization constant, defined as:

$$C = \sum_{v \in D_{i,u}} c(u,v). \tag{7}$$

Note that, in an earlier study [26], we used matrix factorization methods for collaborative filtering on this dataset. However, due to a large number of parameters that must be set properly, slow running time, and comparable accuracy, we preferred to use this simple neighborhood-based collaborative method.

4.3. Hybrid recommendation

Ratings produced by different recommendation methods can be combined with the use of different methods. In order to put the ratings obtained from different methods within the same range, we first do a min-max normalization. This normalization method normalizes the predicted ratings to [0:1] range.

$$r_s^N(u,i) = \frac{r_{s(u,i)} - mR_{u,s}}{MR_{u,s} - mR_{u,s}} \tag{8}$$

In this equation, $r_s(u,i) \subseteq \{r_1(u,i), \ldots, r_k(u,i), r_K(u,i), r_c(u,i)\}$, i.e. $r_s(u,i)$ could be a content-based or collaborative rating, $mR_{u,s}$ indicates the minimum predicted rating of user u calculated according to recommender s in the training set, and $MR_{u,s}$ indicates the maximum predicted rating of user u calculated according to recommender s in the training set.

Ratings can be brought together using a number of different methods. We present three of them here:

- **Content and collaborative hybrid combiner:** This combination method just takes a linear combination of ratings produced by the content-based method and the collaborative method.

$$r_{k,c,a}(u,i) = a \times r_c^N(u,i) + (1-a) \times r_k^N(u,i) \tag{9}$$

- **Sum combiner:** This method produces a combined rating summing normalized ratings $r_s^N(u,i) \subseteq \left\{r_1^N(u,i), \ldots, r_K^N(u,i), r_c^N(u,i), r_{1,c,a}^N(u,i), \ldots r_{K,c,a}(u,i)\right\}$, i.e. we use the normalized content ratings of actor, genre, director, keyword, release year, or collaborative rating or normalized content and collaborative hybrid ratings in the sum.

$$r_{sum}^N(u,i) = \sum r_s^N(u,i) \tag{10}$$

- **Best combiner:** This method evaluates the recommendation methods based on their performance during testing for each user separately. Given a testing day t, we can compute the performance $\theta_s(t^{'},u)$ (see Section 5) of a recommendation method s for user u for a previous testing day $t' < t$. We choose the best recommender according to the cumulative performance from the beginning of testing to the day before the current testing day:

$$\sum_{t^{'}=T_{train}+1\ldots t-1} \theta_s(t^{'},u) \tag{11}$$

where $t^{'} = T_{train}+1 \ldots t-1$ denotes the days between the beginning of testing and the day right before day t. This combination method allows different users to have different recommenders for each day.

Recommendation methods that adapt with time were studied in [27]. In that study, collaborative filtering was treated as a time-dependent and iterative prediction problem. On a Netflix dataset, the current system performance was used to select future neighborhood size, which was used to select and update other parameters. These parameters were tuned for each user and each time period.

In Figure 3, we show a flowchart of all the recommendation methods used in our paper.

5. Evaluation

5.1. Recommender evaluation methodology

We separate all the available data into training and test sets. The training set consists of data on days $t = 1 \ldots T_{train}$ and the test set consists of the data on days $T_{train}+1 \ldots T_{test}$. We use the training set to produce the implicit ratings (Eq. (5)) and content features for each feature set type and to compute the content feature-based user profiles (Eq. (3)). We use the test set to evaluate each recommender.

Since movies need to be recommended to the TV users on a daily basis, we evaluate each recommender for each day of the testing period and take an average over the test days to compute the performance of the recommender for a particular user in the test set. We then take an average over all users in the test set to compute the overall performance of the recommender.

5.2. Recommender evaluation metrics

In order to evaluate the performance of our recommendation methods, we use the traditional methods of evaluation, namely precision, recall, and F-measure metrics.

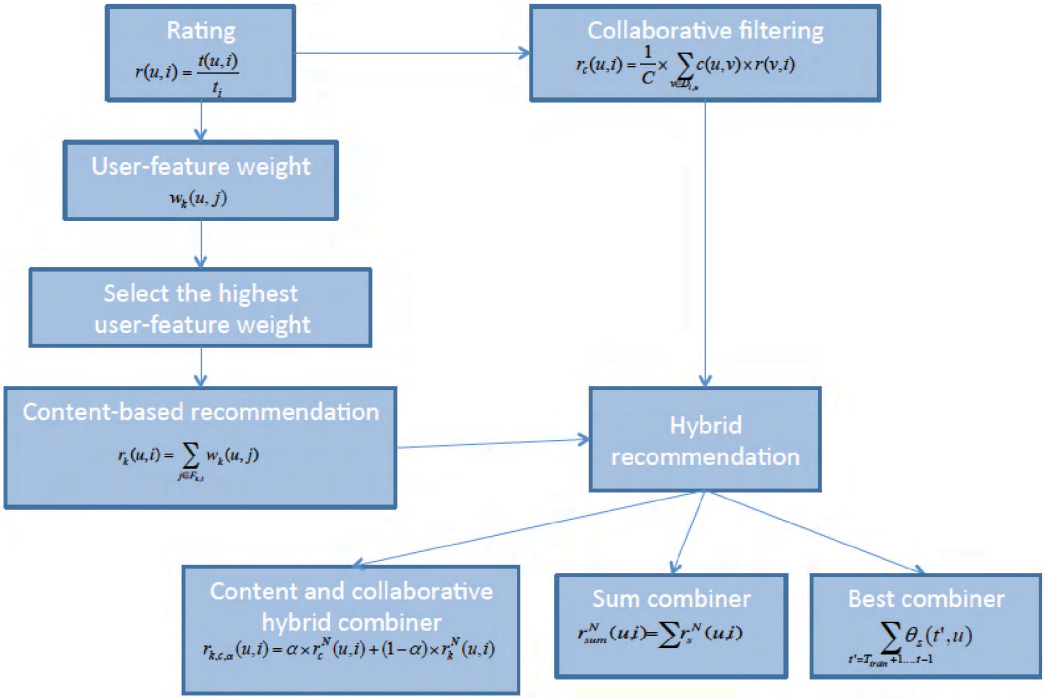

Figure 3. Flowchart of the methods used for movie recommendation.

First of all, we compute the predicted ratings $r_s^N(u,i)$ of the movies on a particular test day in the test set for user u. We sort the predicted ratings in decreasing order. We evaluate the hitCount, the number of movies watched by the user, on *topn* $(n = 3)$ of these ordered movies and then we compute the precision, recall, and F-measure metrics.

Precision is the measure of the how many movies the system hits in the *topn* movies:

$$Precision = \frac{hitCount}{N}. \tag{12}$$

Recall is the ratio of the number of hits out of n recommendations and the size of the number of movies that user u watched in the test set:

$$Recall = \frac{hitCount}{I_u^{test}}. \tag{13}$$

For example, if the recommendation system hits 5 movies that have been watched out of the 10 recommended movies, the precision value will be 0.5. If the user watched 30 movies in the test set, then the recall value will be 0.16. The F-measure (also called F1-measure) combines both precision and recall as follows:

$$F - Measure = 2 \times \frac{Precision \times Recall}{Precision + Recall}. \tag{14}$$

In our experiments, since we recommend movies to a user for each day, we recommend a list of $n = 3$ movies and measure performance based on this list.

In addition to these metrics, we use another performance measure for a TV recommender. This measure, which we call the "EmptyRatio", is a measure of how many times a recommender failed to generate a recommendation for the test period. We think this is an important measure because of the data sparsity in TV movie

recommendation. There may be movies for which there is no related feature in the user profile or neighbor users' profiles. We would like a recommender to be able to recommend something, instead of not being able to produce anything at all.

We define the EmptyRatio metric as follows: Let $b_s(u)$ be the number of movies in test data for which recommender s could not produce a recommendation. Let b_s^- be the average value of $b_s(u)$ over all users, and $mb = min_s b_s^-$ and $Mb = max_s b_s^-$. Then the EmptyRatio for recommender s is defined as:

$$EmptyRatio\,(s) = \frac{b_s^- - mb}{Mb - mb}. \tag{15}$$

6. Experimental results

In this section we provide the experimental results using different types of content feature-based user profiles for recommendation. We also evaluate the performance of feature selection on keyword, director, and actor feature types and recommender combination methods.

In our experiments, we used a dataset consisting of viewing durations of 425 users for 982 movies during a 13-month time interval. Data collected in first 12 months were used for the training phase and the last 1 month for the test phase. To construct the feature set we categorized the movies according to their metadata, namely actors, director, keywords, genre, and release year. To be more specific, the training contents that we have used in our experiments include 5074 actors, 2159 directors, 5223 keywords, and 36 genres and have been categorized according to 6 release year intervals.

6.1. Content-based recommendation results

First of all, we evaluated different content-based recommenders for each feature type, namely actor, director, keyword, genre, and year. We also evaluated the neighborhood-based collaborative method (Section 4.2.). In Table 3, we show the precision, recall and F-measure values for these methods. According to these results, content recommenders based on only actor, director, and keyword features outperform the collaborative method. The recommender based on only actor features gives the best results. Genre- and year-based recommenders perform significantly worse than the other methods (the error bars for the performances in Table 3 are around 0.005). Table 3 also shows that recommenders that perform better according to precision also perform better according to recall and F-measure. Therefore, in the next section, we only report the precision values as performance measures.

Table 3. Performance of content-based and collaborative recommenders.

Method	Precision	Recall	F-measure
Actor	0.135	0.099	0.115
Director	0.127	0.092	0.106
Keyword	0.105	0.077	0.089
Genre	0.077	0.055	0.064
Year	0.064	0.045	0.053
Collaborative	0.099	0.075	0.085

6.2. Feature selection results

According to content-based recommendation results (Table 3), actor, director, and keyword feature-based recommenders performed the best. Therefore, we evaluated feature selection performance on these three types

of features. According to the precision values shown in Table 4, feature selection results depend on the type of the feature set being considered. Feature selection helps a lot when noisy features that affect the performance negatively are removed. Although feature selection does not improve the recommendation accuracy significantly, it is still helpful since it is possible to have the same performance with a reduced set of features, thus saving time and money. We were able to achieve the same performance with a smaller feature subset for the keyword (70%) and actor (40%) features. On the other hand, feature selection does not improve the performance for the director feature set. Therefore, all director features need to be considered.

Table 4. Precision values obtained for different percentages of selected features for different content-based recommenders. Best values are bolded.

Perc. selected features	Actor	Director	Keyword
100	0.135	**0.127**	0.105
90	0.135	0.123	0.105
80	**0.136**	0.123	0.107
70	0.135	0.124	**0.109**
60	0.135	0.122	0.104
50	0.135	0.120	0.104
40	0.135	0.116	0.097
30	0.130	0.112	0.099
20	0.127	0.104	0.103
10	0.103	0.082	0.084

The performance values presented in Table 3 are computed over all users for the testing period. On the other hand, different recommenders could behave differently when the training set size is small or large. For example, in [28], it was found that for collaborative filtering the mean absolute error of recommendations decreased with the increase in the number of rated movies.

In order to determine the recommenders' performance for different training set sizes, we show the average precision values obtained for users who watched a certain number of (1–29, 30–49, 50–99, 100–499, $>$499) movies within the training period.

According to Figure 4, there is a general upward trend for collaborative filtering as the number of movies watched within the training period increases. This finding about the collaborative filtering is in agreement with that reported in [28]. On the other hand, we found that different recommenders may be the optimal choice for different training data points. Figure 4 also shows that as long as there are a couple of hundred movies watched by the user, it does not help knowing about more movies watched by the user. This finding holds for our simple rating model, which could underfit the dataset when there are more training data points. Therefore, when a simple rating model is used, it may be a good idea to discard the old data. Comparing the precision for keyword feature subset and the keyword feature subset with 70% selected features, we see that feature selection helps a lot when the user is new and has watched only tens of movies within the training period. When few movies are watched by a user within the training period, using the selected keyword features is even better than the actor features.

6.3. Recommender combination results

In Section 4.3, we described content collaborative hybrid, sum, and best methods for combination of different recommenders. In this section, we present the evaluation results of these hybrid recommendation methods. Since we evaluated recommenders for different values of α in Eq. (9) and different subsets of recommenders in

the sum and best combination methods, there are many combined recommenders. We only show some of the methods in Table 5.

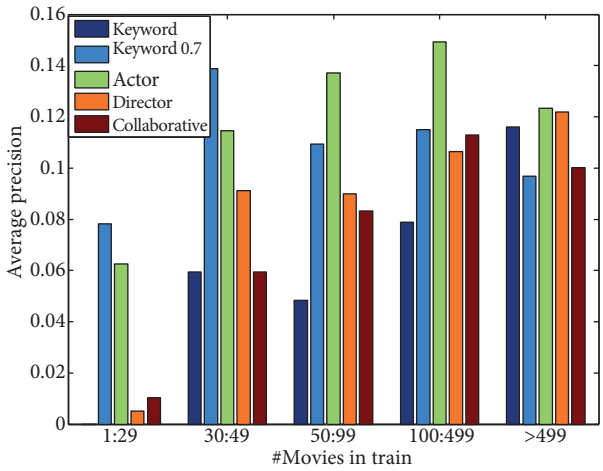

Figure 4. Performance comparison of recommenders on users who watched a certain number of movies within the training period.

Table 5. Performance values obtained for combined recommenders.

Hybrid recommender	Precision	Recall	F-measure	EmptyRatio
Sum(Actor0.8, Director, Keyword0.7)	0.140	0.100	0.116	0.023
Best(Actor0.8, Director, Keyword0.7)	0.138	0.102	0.117	-
Best(Actor, Director, Keyword)	0.138	0.102	0.117	-
Sum(Actor, Director, Keyword)	0.138	0.099	0.115	0
Actor0.8	0.136	0.101	0.116	0.667
Actor	0.135	0.099	0.115	0.592
$0.1 \times$ Collab $+ 0.9 \times$ Actor0.8	0.135	0.097	0.113	0.807

According to Table 5, the sum combination of director and feature selected actor and keyword results in the best performance in terms of precision. It also performs very well in terms of the EmptyRatio criterion, i.e. it is able to produce recommendations for a lot more movies compared to the actor-based content recommender. Since the best method could choose a different recommender for each user and each day, we did not compute the EmptyRatio value for it. For the best combiner, we used the precision as the performance measure of the recommenders in the previous days. The hybrid of actor and collaborative filtering resulted in better performance than collaborative filtering only; however, again this method also has the drawback of a larger EmptyRatio.

7. Conclusions and future work

In this paper, we have devised a method to compute user profiles based on different types of content features (keyword, actor, director, genre, and release year) and implicitly computed movie ratings. We have evaluated a number of content-based recommendation methods using movies' keywords, actors, director, genres, and release year. We obtained the keywords from Turkish movie description documents using natural language processing techniques. We presented a fast filter-type feature selection method for content-based user profiles and presented results of feature selection. We found out that different amounts of feature selection could be

needed for different feature types. Finally, we also provided hybrid recommendation methods that take the sum or the best of available recommender outputs among different recommenders. We have shown that the sum combined recommenders are able to give recommendations for more items. For example, even if the user has not ever seen any of the movie actors before, s/he may have seen a movie by the director and may be interested in the movie. The sum combined recommender is able to make use of whatever information is available in the content.

Movie recommendation, among other fields of recommendation, is a flourishing field and there are many possible future work directions, especially at the intersection of recommendation and machine learning. Zuckerman and Albrecht [29] described statistical models such as neural networks and hidden Markov models, which can be used to learn user profiles. In our initial experiments to learn user profiles using linear models with all the available features, we found out that one needs to be extremely careful about overfitting the few number of movies watched by each user. In this study, we did not use statistical learners for user model but just let the implicit ratings and the existence of features determine the user profiles. We think that after careful feature selection, statistical learning can be used successfully.

We think that especially statistical learning methods that can use labels and features of related users and content on networked data [30] need to be considered. The learning methods for recommenders also need to be able to deal with multiple and different feature dimensions (user, item, and contextual information such as time, place [31], user activities, interests, moods, experiences, and demographic information [32]).

Another research direction is the evaluation methodologies. As more complex models using statistical learning methods are trained, it will be very important how different methods are evaluated so that the best recommender can be found. Methods that can take the user preferences [8,9] or nature of the recommendation scenario into account better are needed. For example, instead of precision, rating weighted normalized precision in [33] or serendipity in [34] could be used as performance measures.

There are many feature selection methods with both different feature evaluation and search methods [18]. They can be considered in conjunction with the recommenders' evaluation methods and user and content clustering to get better results. As mentioned in [35], since users navigate through different social websites (Facebook, YouTube) and different mobile applications, we think that user model interoperability and having user models with as few features as possible will also become very important.

Acknowledgments

This work was supported in part by the Turkish Ministry of Science, Industry and Technology, San-Tez Program, Project number 00966.STZ.2011-2, and in part by Digiturk.

References

[1] Santos da Silva F, Alves LGP, Bressan G. PersonalTVware: an infrastructure to support the context-aware recommendation for personalized digital TV. Int J Comput Inf Sci 2012; 4: 131–135.

[2] Ali K, Van Stam W. TiVo: Making show recommendations using distributed collaborative filtering architecture. In: KDD '04: 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2004; Seattle, WA, USA. New York, NY, USA: ACM. pp. 394–401.

[3] Wang J, de Vries AP, Reinders MJT. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In: 29th ACM SIGIR Conference on Information Retrieval; 2006; Seattle, WA, USA. New York, NY, USA: ACM. pp. 501-508.

[4] Koren Y, Bell R, Volinsky C. Matrix factorization techniques for recommender systems, Comput J 2009; 42: 30–37.

[5] Melville P, Mooney RJ, Nagarajan R. Content-boosted collaborative filtering. In: Proceedings of Eighteenth National Conference on Artificial Intelligence (AAAI-2002); Alberta, Canada. pp 187–192.

[6] Glaser WT, Westergren TB, Stearns JP, Kraft JM. Consumer Item Matching Method and System. U.S. Patent No. 7,003,515. Washington, DC, USA: US Patent and Trademark Office, 2006.

[7] Mooney RJ, Roy, L. Content-based book recommending using learning for text categorization. In: 5th ACM Conference on Digital Libraries; 2000; San Antonio, TX, USA. New York, NY, USA: ACM. pp. 195–204.

[8] Pu P, Chen L, Hu R. Evaluating recommender systems from the users perspective: survey of the state of the art. User Model User-Adap 2012; 22: 317–355.

[9] Konstan JA, Riedl J. Recommender systems: from algorithms to user experience. User Model User-Adap 2012; 22: 101–123.

[10] Burke, R. Hybrid recommender systems: survey and experiments. User Model User-Adap 2002; 12: 331–370.

[11] Goren-Bar D, Glinansky O. FIT-recommending TV programs to family members. Comput Graph 2004; 28: 149–156.

[12] Yu Z, Zhou X, Hao Y, Gu, J. TV program recommendation for multiple viewers based on user profile merging. User Model User-Adap 2006; 16: 63–82.

[13] Ikawa K, Fukuhara T, Fujii H, Takeda H. Evaluation of a TV programs recommendation using the EPG and viewer's log data. In: EuroITV; 2010; Tampere, Finland. New York, NY, USA: ACM. pp. 182–185.

[14] Degemmis M, Lops P, Semeraro G. A content-collaborative recommender that exploits WordNet-based user profiles for neighborhood formation. User Model User-Adap 2007; 17: 217–255.

[15] Bilgin O, Çetinoğlu O, Oflazer K. Building a WordNet for Turkish. Rom J Inf Scı Tech 2004; 7: 163–172.

[16] Madylova A, Gündüz-Öğüdücü S. Comparison of similarity measures for clustering Turkish documents. Intell Data Anal 2009; 13: 815–832.

[17] Ceylan U, Birtürk A. Combining feature weighting and semantic similarity measure for a hybrid movie recommender system. In: Fifth SNAKDD-ACM SIGKDD Workshop on Social Network Mining and Analysis, held in conjunction with the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2011); 21 August 2011; San Diego, CA, USA. New York, NY, USA: ACM. pp. 42–50.

[18] Guyon I, Elisseeff, A. An introduction to variable and feature selection. J Mach Learn Res 2003; 3: 1157–1182.

[19] Uluyağmur M, Çataltepe Z, Tayfur E. Content-based movie recommendation using different feature sets. In: International Conference on Machine Learning and Data Analysis (ICMLDA'12); 2012; San Francisco, CA, USA. Hong Kong: IAENG. pp. 517–521.

[20] Yu Z, Zhou X. TV3P: An adaptive assistant for personalized TV. IEEE T Consum Electr 2004; 50: 393–399.

[21] Akın AA, Akın MD. Zemberek, an open source NLP framework for Turkic languages. Structure 2007; 10: 1–5.

[22] Salton G, Buckley C. Term-weighting approaches in automatic text retrieval. Inform Process Manag 1988; 24: 513–523.

[23] Ding C, Peng H. Minimum redundancy feature selection from microarray gene expression data. In: IEEE 2003 Bioinformatics Conference; 11–14 August 2003; Stanford, CA, USA. New York, NY, USA: IEEE. pp. 523–528.

[24] Şenliol B, Gülgezen G, Yu L, Çataltepe Z. Fast correlation based filter (FCBF) with a different search strategy. In: ISCIS 2008 Conference; 27–29 October 2008; İstanbul, Turkey. New York, NY, USA: IEEE. pp. 1–4.

[25] Cover T, Thomas J. Elements of Information Theory. 2nd ed. New York, NY, USA: Wiley, 1991.

[26] Çataltepe Z, Uluyağmur M, Tayfur E. TV program recommendation using implicit feedback with adaptive regularization. In: IEEE 20th Signal Processing and Communications Applications Conference (SIU); 2012; Muğla, Turkey. New York, NY, USA: IEEE. pp. 18–20 (in Turkish with abstract in English).

[27] Lathia N, Hailes S, Capra, L. Temporal collaborative filtering with adaptive neighbourhoods. In: 32nd international ACM SIGIR Conference on Research and Development in Information Retrieval; 2009; Boston, MA, USA. New York, NY, USA: ACM. pp. 796–797.

[28] Berkovsky S, Kuflik T, Ricci F. Mediation of user models for enhanced personalization in recommender systems. User Model User-Adap 2008; 18: 245–286.

[29] Zukerman I, Albrecht DW. Predictive statistical models for user modeling. User Model User-Adap 2001; 11: 5–18.

[30] Macskassy SA, Provost F. Classification in networked data: A toolkit and a univariate case study. J Mach Learn Res 2007; 8: 935–983.

[31] Adomavicius G, Sankaranarayanan R, Sen S, Tuzhilin A. Incorporating contextual information in recommender systems using a multidimensional approach. ACM T Inform Syst 2005; 23: 103–145.

[32] Hsu SH, Wen M, Lin H, Lee CC, Lee CH. AIMED-A personalized TV recommendation system. In: 5th European Conference, EuroTV; 2007; Amsterdam, the Netherlands. Berlin, Germany: Springer. pp. 166–174.

[33] Uluyağmur M. Hybrid movie recommendation. MSc, İstanbul Technical University, İstanbul, Turkey, 2012 (in Turkish with abstract in English).

[34] Murakami T, Mori K, Orihara R. Metrics for evaluating the serendipity of recommendation lists. In: JSAI Conference and Workshops; 18–22 June 2007; Miyazaki, Japan. Berlin, Germany: Springer. pp. 40–46.

[35] Carmagnola F, Federica Cena F, Gena C. User model interoperability: a survey. User Model User-Adap 2011; 21: 285–331.