In Proceedings AAAI-98

Recommendation as Classification: Using Social and Content-Based Information in Recommendation

Chumki Basu*
Bell Communications Research
445 South Street
Morristown, NJ 07960-6438
cbasu@bellcore.com

Haym Hirsh
Department of Computer Science
Rutgers University
Piscataway, NJ 08855
hirsh@cs.rutgers.edu

William Cohen
AT&T Laboratories
180 Park Ave, Room A207
Florham Park, NJ 07932
wcohen@research.att.com

Abstract

Recommendation systems make suggestions about artifacts to a user. For instance, they may predict whether a user would be interested in seeing a particular movie. Social recomendation methods collect ratings of artifacts from many individuals and use nearest-neighbor techniques to make recommendations to a user concerning new artifacts. However, these methods do not use the significant amount of other information that is often available about the nature of each artifact — such as cast lists or movie reviews, for example. This paper presents an inductive learning approach to recommendation that is able to use both ratings information and other forms of information about each artifact in predicting user preferences. We show that our method outperforms an existing social-filtering method in the domain of movie recommendations on a dataset of more than 45,000 movie ratings collected from a community of over 250 users.

Introduction

Recommendations are a part of everyday life. We usually rely on some external knowledge to make informed decisions about a particular artifact or action, for instance when we are going to see a movie or going to see a doctor. This knowledge can be derived from social processes. At other times, our judgments may be based on available information about an artifact and our known preferences. There are many factors which may influence a person in making choices, and ideally one would like to model as many of these factors as possible in a recommendation system.

There are some general approaches to this problem. In one approach, the user of the system provides ratings of some artifacts or items. The system makes informed guesses about other items the user may like based on ratings other users have provided. This is the framework for *social-filtering* methods (Hill, Stead, Rosenstein & Furnas 1995; Shardanand & Maes 1995). In a second approach, the system accepts information describing the nature of an item, and based on a sample of the user's preferences, learns to predict which items the user will like (Lang 1995; Pazzani, Muramatsu, & Billsus 1996). We will call this approach *content-based filtering*, as it does not rely on social information (in the form of other users' ratings). Both social and content-based filtering can be cast as learning problems: the objective is to learn a function that can take a description of a user and an artifact and predict the user's preferences concerning the artifact.

Well-known recommendation systems like *Recommender* (Hill, Stead, Rosenstein & Furnas 1995) and *Firefly* (http: //www.firefly.net) (Shardanand & Maes 1995) are based on social-filtering principles. *Recommender*, the baseline system used in the work reported here, recommends as yet unseen movies to a user based on his prior ratings of movies and their similarity to the ratings of other users. Social-filtering systems perform well using only numeric assessments of worth, i.e., ratings. However, social-filtering methods leave open the question of what role content can play in the recommendation process.

Let's take the case of movie data. On the Web alone, one is likely to find the following information for a given movie: a breakdown of cast/crew, plot, movie production details, reviews, trailer, film and audio clips, (and ratings too). When deciding on a movie to see, a person has easy access to this material. Social-filtering may be characterized as a generic approach, unbiased by the regularities exhibited by properties associated with the items of interest (Hill, Stead, Rosenstein & Furnas 1995). (A significant motivation for some of the work on such systems is to explore the utility of recognizing communities of users based solely on similarities in their preferences.) However, the fact that content-based properties can be identified at low cost (with no additional user effort) and that people are influenced by these regularities make a compelling reason to investigate how best to use them.

When are ratings alone insufficient? Social-filtering makes sense when there are enough other users known to the system with overlapping characteristics. It is dependent upon the current state of the system — the number of users and the number and selection of movies that have been rated. So, for example, when a new movie comes out, there will be a period of time when a recommendation system will have little ratings data for this movie, thus affecting its ability to make reliable recommendations for this movie.

In this paper, we present a new, inductive learning approach to recommendation. We show how pure social-filtering can be accomplished using this approach, how the naive introduction of content-based information does not help — and indeed harms — the recommendation process, and finally, how the use of hybrid features that combine elements of social and content-based information makes it possible to achieve more accurate

*Department of Computer Science, Rutgers University, Piscataway, NJ 08855
We would like to thank Susan Dumais for useful discussions during the early stages of this work.

recommendations. We use the problem of movie recommendation as our exploratory domain for this work since it provides a domain with a large amount of data (over 45,000 movie evaluations across more than 250 people), as well as a baseline social-filtering method to which we can compare our results (Hill, Stead, Rosenstein & Furnas 1995).

The Movie Recommendation Problem

In the social-filtering approach, a recommendation system is given as input a set of ratings of specific artifacts for a particular user. In movie recommendation, for instance, this input would be a set of movies that the user had seen, with some numerical rating associated with each of these movies. The output of the recommendation system is another set of artifacts, not yet rated by the user, which the recommendation system predicts the user will rate highly.

Social-filtering systems would solve this problem by focusing solely on the movie ratings for each user, and by computing from these ratings a function that can give a rating to a user for a movie that others have rated but the user has not. These systems have traditionally output ratings for movies, rather than a binary label. For example, *Recommender* computes for a user a smaller group of reference users, known as recommenders, who are other community members most similar to the user. Using regression techniques, these recommenders' ratings are used to predict ratings for new movies.

Content-based recommendation systems, on the other hand, would reflect solely the non-ratings information. For each user they would take a description of each liked and disliked movie, and learn a procedure that would take the description of a new movie and predict whether it will be liked or disliked by the user. For each user a separate recommendation procedure would be used.

Our Approach

Our goal is to develop an approach to recommendation that can exploit both ratings and content information. We depart from the traditional social-filtering approach by framing the problem as one of classification, rather than artifact rating. However, we differ from content-based filtering methods in that social information, i.e. other users' ratings, will be used in the inductive learning process.

We will formalize the movie recommendation problem as a learning problem—specifically, the problem of learning a function that takes as input a user and a movie and produces as output a label indicating whether the movie would be liked (and recommended) or disliked:

$$f(\langle user, movie \rangle) \rightarrow \{liked, disliked\}$$

As a classification problem, we are also interested in predicting whether a movie is liked or disliked, not an exact rating. Our output is not an ordered list of movies, but a set of movies which we predict will be liked by the user. Most importantly, we are now able to generalize our inputs to the problem to other information describing both users and movies.

The information we have available is a collection of user/movie ratings (on a scale of 1-10), and certain additional information concerning each movie.[1] To present the results as sets of movies predicted to be liked or disliked by a user we compute a ratings threshold for each user such that 1/4 of all the user's ratings exceed it and the remaining 3/4 do not, and we return as recommended any movie whose predicted rating is above the training-data-based threshold on movies.

Below we will outline a number of alternative ways that a user/movie rating might be represented for the learning system. We will first describe how we represent social recommendation information, which we call "collaborative" features, then how we represent "content" features, and finally describe the hybrid features that form the basis for our most successful recommendation system.

Collaborative Features

As an initial representation, we use a set of features that takes into account, separately, user characteristics and movie characteristics. For instance, perhaps a group of users were identified as liking a specific movie:

> Mary, Bob, and Jill liked *Titanic*.

We grouped users like these into a single feature called *users who liked movie X*, the value of which is a set. (*E.g.*, { Mary, Bob, Jill} would be the value of the feature *users who liked movie X* for the movie *Titanic*). Since our ground ratings data contain numerical ratings, we say a user likes a movie if it is rated in the top-quartile of all movies rated by that user.[2]

We also found it important to keep track of the user's favorite movies, namely those which appeared in his top-quartile:

> Tim liked the movies, *Twister*, *Eraser*, and *Face/Off*.

We developed an attribute, *movies liked by user*, which encoded this information. Collectively, we call these attributes *collaborative features* since they used data known to social-filtering systems: users, movies, and ratings.

As a result, every user/movie rating is converted into a tuple of two set-valued features. The first attribute is a set containing the movies liked by the user, and can be thought of as a single attribute describing the user. The second attribute is a set containing the users who like the given movie, and can be thought of as a single attribute describing the movie. Each such tuple is labeled by whether it was liked or disliked by the user, according to whether it was in the top-quartile for the user.

The use of set-valued features led naturally to use of *Ripper*. *Ripper* is an inductive learning system that is able to learn rules from data with set-valued attributes (Cohen 1995; 1996). A rule contains a conjunction of several tests. In the case of a set-valued feature f, a

[1] It would be desirable to make the recommendation process a function of user attributes such as age or gender, but since that information is not available in the data we are using in this paper, we are forced to neglect it here.

[2] The value of 1/4 was chosen rather arbitrarily, and our results are similar when this value was changed to 20% or 30%.

test may be of the form "$e_i \in f$" where e_i is some constant that is an element of f in some example. As an example, *Ripper* might learn a rule containing the test *Jaws* $\in$ *movies-liked-by-user*.

Content Features

Content features are more naturally available in a form suitable for learning, since much of the information concerning a movie are available from (semi-) structured online repositories of information. An example of such a resource which we found very useful for movie recommendation is the Internet Movie Database (IMDb) (http: //www.imdb.com). The following content features in our experiments were extracted from this database: Actors, Actresses, Directors, Writers, Producers, Production Designers, Production Companies, Editors, Cinematographers, Composers, Costume Designers, Genres, Genre Keywords, User-submitted Keywords, Words in Title, Aka (also-known-as) Titles, Taglines, MPAA rating, MPAA reason for rating, Language, Country, Locations, Color, Soundmix, Running Times, and Special Effects Companies.

Hybrid Features

Our final set of features reflect the common human-engineering effort that involves inventing good features to enable successful learning. We looked for content that was frequently associated with the movies in our data and that is often used when choosing a movie. An example would be a movie's *genre*. To make effective use of the *genre* feature, we needed to relax an apparently natural assumption: that a $\langle user, movie \rangle$ pair would be encoded as a set of collaborative features, plus a set of content features describing the movie. Instead, we found it more effective to define new collaborative features that are influenced by content, which we call *hybrid features*.

We isolated three of the most frequently occurring genres in our data — comedy, drama, and action. We then introduced features that isolated groups of users who liked movies of the same genre, such as *users who liked dramas*. Similar features were defined for comedy and action movies. These features combine knowledge about users who liked a set of movies with knowledge of some content associated with the movies in the set. Definitions concerning what it means for a user to like a movie remain the same (top-quartile) as in earlier parts of this paper.

Experiments and Results

We report on some of the significant results of our experiments using different sets of features.

Training and Test Data

Our data set consists of more than 45,000 movie ratings collected from approximately 260 users which originated from a data set that was used to evaluate *Recommender*. However, over the course of our work we discovered that the training and test distributions in this data were distributed very differently. We therefore generated a new partition of data into a training set which contained 90% of the data and a testing set which contained the remaining 10%, for which the two distributions would be more similar. Unfortunately, for some of the users *Recommender* failed to run correctly, and those few users were dropped from this study. Note that this was the only reason for dropping users. No users were dropped due to the performance of our own methods.

We generated a testing set by taking a *stratified random sample* (Moore 1985) of the data as follows:

- For every user, separate and group his movie/rating pairs into intervals defined by the ratings. Movies are rated on a scale from 1 to 10.
- For each interval, take a random sample of 10% of the data and combine the results.

We have clearly defined intervals where all the units in an interval share a common property, the rating. Therefore, the holdout set we computed is more representative of the distribution of ratings for the entire data set than it would have been if we had used simple random sampling.

Evaluation Criteria

Our evaluating a movie as being liked if it is in the top-quartile reflects our belief that knowing the actual rating of a movie is not as important as knowing where the rating was relative to other ratings for a given user. We are really interested in predicting whether a movie would be amongst the user's favorites. This has the nice effect of dealing with the fact that the intervals on the ratings scale are *not equidistant*. For instance, given a scale of 1 to 10 where 1 indicates low preference and 10, high preference, the "qualitative" difference between a rating of 1 and a rating of 2 is less when compared to the difference between 6 and 7, for any user whose ratings are mostly 7 and above.

We rely on two metrics commonly used in information retrieval — *precision* and *recall*. Precision gives us an estimate of how many of the movies predicted to be in the top-quartile for a user really belong to that group. Recall estimates how many of all the movies in the user's top-quartile were predicted correctly. We feel that when recommending movies, the user is more interested in examining a small set of recommended movies rather than a long list of candidates. Unlike document retrieval, where the user can narrow a list of retrieved items by actually reading some of the documents, here, the user is really interested in seeing just one movie. Therefore, our objective for movie recommendation is to maximize precision without letting recall drop below a specified limit. Precision represents the fact that a movie selected from the returned set will be liked, and the recall cutoff reflects the fact that there should be a non-trivial number of movies returned (for example, in case a video store is out of some of the recommended titles).

Baseline Results

In our initial experiment, we applied *Recommender*'s social-filtering methods to compute ratings for the holdout movies. For every individual, we separated his data from the holdout set. The rest of his data along with all of the other users' data was made available to *Recommender*'s analysis routines and a rating was computed for each of his holdout movies.

To determine whether a computed rating is in the top-quartile, we precompute *thresholds* for every user corresponding to the ratings which separate the top from the lower quartiles. To convert a rating, we use the rule:

- If a *predicted rating* >= *user's threshold*, set the rating to "+".
- Otherwise, set the rating to "−".

The thresholds are set individually for each user, using only the training data ratings for the training data threshold, but the full set of data for a user is used to set the testing data threshold.

Our precision estimates are *microaveraged* (Lewis 1991). which meant that our prediction decisions were made from a single group and an overall precision estimate was computed. As shown in Table 1, *Recommender* achieved microaveraged values of 78% for precision and 33% for recall.

Inductive Learning Results

In the first of our inductive learning recommendation experiments using *Ripper*, we represent every data point in the training and holdout sets as a collaborative feature vector. The collaborative features we used were:

- Users who liked the movie
- Users who disliked the movie
- Movies liked by the user

The entire training set and holdout sets are made available to *Ripper* in two separate files. We ran *Ripper* on this data and generated a classification for each example in the holdout set. *Ripper* produces rules that it learns for this data which it uses to make predictions about the class of an example.

The *Ripper* parameters we found most useful in adjusting from the default settings allow *negative tests in set-valued attributes* and varying the *loss ratio*. The first parameter allows the tests in rules to check for non-containment of attribute values within a set-valued feature. (*E.g.*, tests like *Jaws* $\notin$ *movies-liked-by-user* are allowed.) The *loss ratio* is the ratio of the perceived cost of a false positive to the cost of a false negative; increasing this parameter encourages *Ripper* to improve precision, generally at the expense of recall. In most of the experiments, we varied the loss ratio until we achieved a high value of precision with a reasonable recall. At a loss ratio of 1.9, we achieved a microaveraged precision of 77% and a recall of 27% (see Table 1). This level of precision is comparable to *Recommender*, but at a lower level of recall.

In the second set of experiments, we replaced the collaborative feature vectors with a new set of features. Here, we took the 26 features extracted from the IMDb (listed earlier) and added them to the list of collaborative features. We were not able to improve precision and recall at the same time (see Table 1). Recalling that high precision was more important to us than high recall, we find these results generally inferior to that of *Recommender*. Furthermore, examining the rules that *Ripper* generated, we found that content features were seldom used. Based on this experiment, the collaborative data appear to be better predictors of user preferences than our initial encoding of content. In addition, given the high dimensionality of our feature space, it appears to be difficult to make reasonable associations amongst the examples in our problem.

Method	*Precision*	*Recall*
Recommender	78%	33%
Ripper (no content)	77%	27%
Ripper (simple content)	73%	33%
Ripper (hybrid features)	83%	34%

Table 1: Results of the different recommendation approaches.

Next, we created features that combined collaborative with content information relating to the genre of a movie. These hybrid features were:

- Comedies liked by user
- Dramas liked by user
- Action movies liked by user

Although the movies in our data set are not limited to these three genres, we took a conservative approach to adding new features and began with the most popular genres as determined by the data.

To introduce the next set of collaborative features, we face a new issue. For example, we want a feature to represent the set of users who liked comedies. Although we have defined what it means to like a movie, we have not defined what it means to like movies of a particular genre. How many of the movies in the user's top-quartile need to be of a particular genre in order for the user to like movies of that genre?

Based on the proportion of movies for a particular genre in a user's top-quartile, we identified broad clusters. As a first cut, we divided the proportions of movies of different genres into four groups. For each of the popular genres, *comedy*, *drama*, and *action*, we then defined the following features:

- Users who liked many movies of genre X
- Users who liked some movies of genre X
- Users who liked few movies of genre X
- Users who disliked movies of genre X

We also add features including, for example, the genre of a particular movie. Running *Ripper* on this data with a loss ratio of 1.5, we achieved a microaveraged precision of 83% with a recall of 34%. These results are summarized in Table 1.

Using the standard test for a difference in proportions (Mendenhall, Scheaffer, & Wackerly 1981, pages 311-315) it can be determined that *Ripper* with hybrid features attains a statistically significant improvement over the baseline *Recommender* system with respect to precision ($z = 2.25$, $p > 0.97$), while maintaining a statistically indistinguishable level of recall.[3] *Ripper* with hybrid features also attains a statistically significant improvement over *Ripper* without content features with respect to both precision ($z = 2.61$, $p > 0.99$) and recall ($z = 2.61$, $p > 0.998$).

Observations

Our results indicate that an inductive approach to learning how to recommend can perform reasonably well

[3] More precisely, one can be highly confident that there is no practically important loss in recall relative to the baseline; with confidence 95%, the recall rate for *Ripper* with hybrid features is at least 32.8%.

when compared to social-filtering methods, evaluated on the same data. We have also shown that by formulating recommendation as a problem in classification, we are able to combine meaningfully information from multiple sources, from ratings to content. At equal levels of recall, our evaluation criteria would favor results with higher precision. Our results using hybrid features show that even with high precision, we also have a slight edge over recall as well.

Related Work

(Karunanithi & Alspector 1996) compare *clique-based* and *feature-based* models for movie selection. A clique is a set of users whose movie ratings are similar, comparable to the set of recommenders in (Hill, Stead, Rosenstein & Furnas 1995). Those members of the clique who have rated a movie that the user has not seen predict a rating for that movie. Once a clique is formed for a user, a movie rating is estimated by calculating the arithmetic mean of the ratings for the members of the clique. In the feature-based approach, features are extracted from movies a user has rated, a neural-network user model is built associating the features and the ratings, and ratings for unseen movies are computed by considering their features as new inputs to the model. The six features used were a movie's category (genre), Maltin (critic) rating, MPAA rating, Academy Award information, length of movie, and country of origin. The authors found that by using features, in most cases, they outperformed a human critic but almost consistently did worse than the clique method. Although our initial results with content features support these findings, we also demonstrated that content information can lead to improved recommendations, if encoded in an appropriate manner.

Fab (Balabanovic & Shoham 1997) is a system which tackles both issues of content-based filtering and social-filtering. In the *Fab* system, content information is maintained by two types of agents: *user agents* associated with individuals and *collection agents* associated with sets of documents. Each collection agent represents a different topic of interest. Each of these agent-types maintains its own profiles, consisting of terms extracted from documents, and uses these profiles to filter new documents. These profiles are reinforced over time with user feedback, in the form of ratings, for new documents. Among the differences in our approach, ours is not an agent-based framework and we do not have access to topics of interest information, which in *Fab* were collected from the users.

In addition to *Recommender*, another well known social-filtering system is *Firefly*, a descendant of *Ringo* (Shardanand & Maes 1995). (*Firefly* has since expanded beyond the domain of music recommendation). *Ringo* presents the user with a list of artists and albums to rate. This system maintains a dynamic profile of each user's likes and dislikes. Profiles are compared with one another to locate other individuals with similar tastes. A number of similarity metrics are presented, such as *mean-squared difference* and the *Pearson-r measure*. Once the most similar profiles are selected for a user, *Ringo* makes predictions by computing a weighted average of the ratings in these profiles.

Final Remarks

In this paper, we have presented an inductive approach to recommendation and evaluated it via experiments on a large, realistic set of ratings. An advantage of the inductive approach, relative to other social-filtering methods, is its flexibility: we can encode collaborative and content information as part of the problem representation without any algorithmic modifications. Exploiting this flexibility, we have evaluated a number of representations for recommendation, including two types of representations using content. One of these representations, based on hybrid features, significantly improves performance over the purely collaborative approach. We have thus begun to realize the impact of multiple information sources, including sources that exploit a limited amount of content. We believe that this work provides a basis for further work in this area, particularly in harnessing other types of information content.

References

Balabanovic, M.; and Shoham Y. 1997. Content-Based, Collaborative Recommendation. *Communications of the ACM* Vol. 40, No. 3. March, 1997.

Cohen, W. 1995. Fast Effective Rule Induction. In *Machine Learning: Proceedings of the Twelfth International Conference*. Lake Tahoe, California: Morgan Kaufmann.

Cohen, W. 1996. Learning Trees and Rules with Set-valued Features. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*. Portland, Oregon.

Hill, W.;Stead, L.;Rosenstein, M.; and Furnas, G. 1995. Recommending and Evaluating Choices in a Virtual Community of Use. In *Proceedings of the CHI-95 Conference*. Denver, Colorado.

Karunanithi, N.; and Alspector, J. 1996. Feature-Based and Clique-Based User Models for Movie Selection. In *Proceedings of the Fifth International Conference on User Modeling*. Kailua-Kona, Hawaii.

Lang, K. 1995. NewsWeeder: Learning to filter netnews. In *Machine Learning: Proceedings of the Twelfth International Conference*. Lake Tahoe, California: Morgan Kaufmann.

Lewis, D. 1991. Evaluating Text Categorization. In *Proceedings of the Speech and Natural Language Workshop*. Asilomar, California.

Mendenhall, W.; Scheaffer, R.; and Wackerly, D., eds. 1981. *Mathematical Statistics with Applications*. Duxbury Press, second edition.

Moore, D. 1985. *Statistics: concepts and controversies*. W. H. Freeman.

Pazzani, M.; Muramatsu, J.; and Billsus, D. 1996. Syskill & Webert: identifying interesting web sites. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*. Portland, Oregon.

Shardanand, U.; and Maes, P. 1995. Social Information Filtering: Algorithms for Automating "Word of Mouth". In *Proceedings of the CHI-95 Conference*. Denver, Colorado.