A Content-Based Approach to Collaborative Filtering

CS 470: Applied Software Development Project
Brandon Douthit-Wood
April 26, 2004

Keywords: collaborative filtering, recommender system, information retrieval.

Abstract

Collaborative filtering has become a popular method for delivering recommendations to individuals on a wide range of items, most typically books, movies, music and news articles. The basic idea behind collaborative filtering is to automate word-of-mouth. We all rely on friends and family to recommend books, movies and music to us; collaborative filtering aims to expand each person's small network of friends to the greater realm of the Internet. Collaborative filtering works by finding individuals with similar tastes, and making recommendations based on their likes and dislikes. This relies on the idea that if two individuals have similar tastes on a number of items, they are likely to have similar tastes on other items as well. However, there are a number of problems with typical collaborative filtering methods, which can lead to poor recommendations in many situations. This paper presents a method of combining typical collaborative filtering techniques with content-based analysis of the items in order to provide accurate recommendations for a wide range of situations.

1.0 Introduction

Collaborative filtering has developed as a means to create accurate recommender systems. Recommender systems allow individuals to receive recommendations on a wide range of items, including books, movies, music and news articles. Ratings data is collected from a large group of users for a given set of items. These ratings can be collected either explicitly by asking a user to rate an item they are familiar with (on a 1-5 scale, for example), or implicitly by inferring a user's likes and dislikes based on their behavior. Many online stores, such as Amazon and Barnes & Noble, collect ratings implicitly; when a user buys an item, it is assumed that they like that item. By comparing a user's likes and dislikes to other users, groups of users with similar tastes are formed, called a neighborhood. To determine the predicted rating of an item for a particular user, the (possibly weighted) average rating of that item is calculated from the user's neighborhood.

In the past, collaborative filtering systems have been able to produce accurate recommendations for a wide range of items. However, several problems cannot be overcome by a typical collaborative filtering system:

- *Performance* – The data set for a typical collaborative filtering system is prohibitively large. Millions of users, each with possibly hundreds or thousands of ratings, cannot be exhaustively searched by any real-time system.

- *Sparsity of data* – There are actually two problems at issue here. First, there is the problem of a new item that has few (or no) ratings. With pure collaborative filtering, an item must have many ratings in order to calculate an accurate average. Second, there is the problem of a new user who has rated only a few (or no) items; this is known as the first-rater problem. In order to calculate a user's neighborhood, the user must have already rated a number of items. Some recommender systems [9] have solved this problem by requiring users to rate a set number of items (usually around 20) before being able to receive recommendations. While this is an effective solution to the problem, it is inconvenient for the user.

- *Comparing users* – In order to calculate the similarity between users, they must share ratings of common items. If they do not, there is no way of comparing the users and their calculated similarity will be zero. An example illustrating this problem is given below in Table 1.

	User 1	User 2
Billy Madison	4	
Happy Gilmore		5
Mr. Deeds		4
50 First Dates	5	
Big Daddy		4

Table 1: User profiles of movies rated

Table 1 shows the profiles of two users, with the movies that each has rated on a 1-5 scale. Since the two users have not rated any of the same items, a typical collaborative filtering system will not classify them as neighbors. However, it is clear that both users enjoy Adam Sandler movies. Collaborative filtering alone cannot discover this similarity between the two users, and instead views them as having no similarity.

A content-based approach to collaborative filtering is able to overcome these problems. By analyzing the content information of an item, it is possible to deliver accurate recommendations for items with few ratings, and for users that have only rated a few items. Additionally, it is possible to compare users that have no common ratings. This paper will discuss a method of combining classic collaborative filtering techniques with content analysis of the items in order to deliver more robust and accurate recommendations to users. Finally, this paper will discuss several strategies for improving the performance of the proposed methods.

2.0 Data Source

The effectiveness of the proposed method is demonstrated by a movie recommendation system. The rating data used for this system has been obtained from the EachMovie project [1]. Content information for each movie is obtained from the Internet Movie Database (IMDb) [2]. The following information for each movie is collected into a local database: actors, director, plot description and genre.

2.1 EachMovie Dataset

The EachMovie project was conducted by the Compaq Systems Research Center over an 18-month period from 1996-97. During this time, a large dataset of user-movie ratings was collected, consisting of 2,811,983 ratings for 1,628 movies from 72,916 users. The dataset contains basic information for each movie, including title, genre, release year and the IMDb URL. For each user, optional demographic information is provided such as age, gender and zip code; the demographic information is not used by this system. Finally, the dataset contains the movie rating data for each user. The ratings are given on a one-to-five scale. A rating of one represents the lowest possible score for a movie, while a rating of five is the highest.

For the purposes of this study, the ratings are converted to either a positive or a negative rating. A rating between three and five is considered positive, while a rating of one or two is considered negative. When the predicted ratings are generated, they are calculated simply as a positive or a negative rating.

Unlike other systems that use only a subset of this data [3], the entire dataset is used for testing the effectiveness of this system. The ratings data is randomly split into a training and test set; the training set contains approximately 75 percent of the ratings, while the test set contains the remaining 25 percent.

2.2 Internet Movie Database

The content information for each movie is collected from the Internet Movie Database (IMDb). Information for each movie, such as actors, director, plot summary and genre, are gathered and entered into a local database. This data is provided in tab-delimited text files, which are parsed to collect this information for each movie from the EachMovie dataset.

2.3 Creating the Feature List

Once the content information is gathered, a feature list is constructed in the form of a bag-of-words; common words with little contextual information (i.e.: the, and, but) are thrown out. For the remaining words, the frequency of each is calculated, and any words with a low frequency (only one occurrence) are also thrown out. An example of this bag-of-words representation is shown below in Table 2 for the movie *Goldeneye.*

Goldeneye			
satellite	2	destroy	2
xenia	3	london	2
thriller	2	villain	2
simon	4	revenge	2

Table 2: Bag-of-words representation of movie content information

Before a movie's content information is added to the local database, the frequency of each word is weighted based on the total number of terms in the bag-of-words.

Each user is assigned two feature lists, one for the movies they have rated positively, and another for the movies they have rated negatively. For each movie that a user has rated positively, the feature list of that movie is added to the user's positive feature list. Similarly, the feature list for each movie a user has rated negatively is added to the user's negative feature list.

3.0 Methodology

It is possible to overcome many of the limitations of collaborative filtering methods by combining them with content-based analysis. However, overall system performance remains problematic. Three methods are attempted by this study with the aim of addressing this problem. The first two attempt to improve performance by reducing the search space. The third circumvents collaborative filtering methods altogether by making purely content-based predictions.

3.1 Clustering of Users

It is possible to reduce the search space significantly by pre-processing the ratings data in order to cluster users into smaller, more manageable groups. When the predictions are to be made, the system must only analyze a user's cluster group, instead of the entire data set. Clustering of users (and/or items) has been implemented by several collaborative filtering systems [5,6] as a method of improving performance.

For this study, a relatively simple clustering algorithm was devised:

1. Compare the current user to each of the existing clusters (if any) using the Pearson Correlation Coefficient [7].
2. If the similarity between the user and a cluster is above a pre-defined threshold, merge the user into that cluster.
3. Compare the current user to each of the other users using the Pearson Correlation Coefficient.
4. If the similarity between users is above a pre-defined threshold, create a new cluster from the users.
5. If the similarity threshold is never exceeded, create a new cluster with the user having the greatest correlation.
6. Repeat steps 1-5 until all users have been clustered.

It is important to note that with this content-based approach, the terms found in a user's feature list (or the feature list of a cluster) are what is being used to compute the Pearson Correlation Coefficient. The feature list of a cluster is comprehensive of the feature lists of all its users.

Obviously, this is not the most efficient solution possible; the worst-case runtime is $O(n^2)$. However, it is not necessary for this algorithm to be run every time a new rating is added; small changes in the rating data should not affect the structure of the clusters greatly. Instead, the algorithm could be run on a nightly basis to reorganize the clusters. Furthermore, the focus of this study is not the development of an efficient clustering algorithm. Thus, the described algorithm is sufficient for the implementation of this system.

3.1.1 Making Predictions

Once the cluster groups are formed, the system must only search a user's cluster group in order to make predictions for that user. From a user's cluster group, their 10-nearest neighbors (called their neighborhood) are calculated, again using the Pearson Correlation Coefficient. The predicted rating for a particular item is the average rating of that item from the user's neighborhood.

3.2 Selecting a Random Group

A simpler method for reducing the search space is to select a random group of users that will serve as a (hopefully) representative sample of the entire set of users. For this study, 5000 users are randomly selected to serve as the sample group. A new random group is generated to make predictions for each user.

This is a much simpler and easier method to implement than was the clustering of users. The questions that remain are how representative a sample of 5000 random users will be, and how accurate of predictions is that group able to produce?

3.2.1 Making Predictions

Once a random group of users is selected, the method of prediction is identical to that used with the cluster groups. From a user's random group, their 10-nearest neighbors are calculated, and the predicted rating for a particular item is the average rating of that item from the user's neighborhood.

3.3 Content-Based Predictions

The final method of prediction is the most simple of the three. By completely circumventing the methods of collaborative filtering, predicted ratings are made by simply analyzing the content information of the items a user has rated. Just as we had previously compared feature lists between users, it is also possible to compare the feature list of a user directly to the feature list of an item.

Remember that each user has both a positive and a negative feature list. Each of these feature lists is compared to the feature list of the item using the Pearson Correlation Coefficient. If the user's positive feature list has a higher correlation with the item's feature list, the system predicts a positive rating. Likewise, the system predicts a negative rating if the negative feature list is more highly correlated.

4.0 Experimental Results

Once the predicted ratings were made by each of the three described methods, those predictions were analyzed to assess the quality of the prediction method.

4.1 Metrics

For each of the prediction methods, three metrics were calculated to analyze the quality of the predictions: accuracy, precision and recall. Accuracy is an overall measure of the number of correct predictions made, both positive and negative. Precision is defined as the number of correct positive predictions divided by the number of correct and incorrect positive predictions. Finally, recall is a measure of how well the system is able to generalize. If the system simply memorizes the training set, it would be unable to generalize very well, and will not produce good predictions for the test set. The recall in this case would be very low. It is important to note that precision and recall have an inverse relationship; it is possible to do very well in one or the other, but not both. The desired result is to find a middle ground, where both precision and recall are reasonably good, but neither is outstanding.

In order to calculate the three metrics, the predicted rating was compared to the actual rating for each of the ratings in the test set. The actual calculations used to determine accuracy, precision and recall are shown below in Figure 1.

Accuracy = (TP + TN) / (TN + TP + FN + FP)
Precision = TP / (TP + FP)
Recall = TP / (TP + FN)

TP: number of true positive predictions
TN: number of true negative predictions
FP: number of false positive predictions
FN: number of false negative predictions

Figure 1: Calculations for accuracy, precision and recall

4.2 Results

The results of these experiments are summarized below in Figure 2. As can be seen, the user clustering method performed higher than the other methods on all three metrics. Each of the other two methods produced very similar results, with the random user groups performing slightly better than the purely content-based predictions.

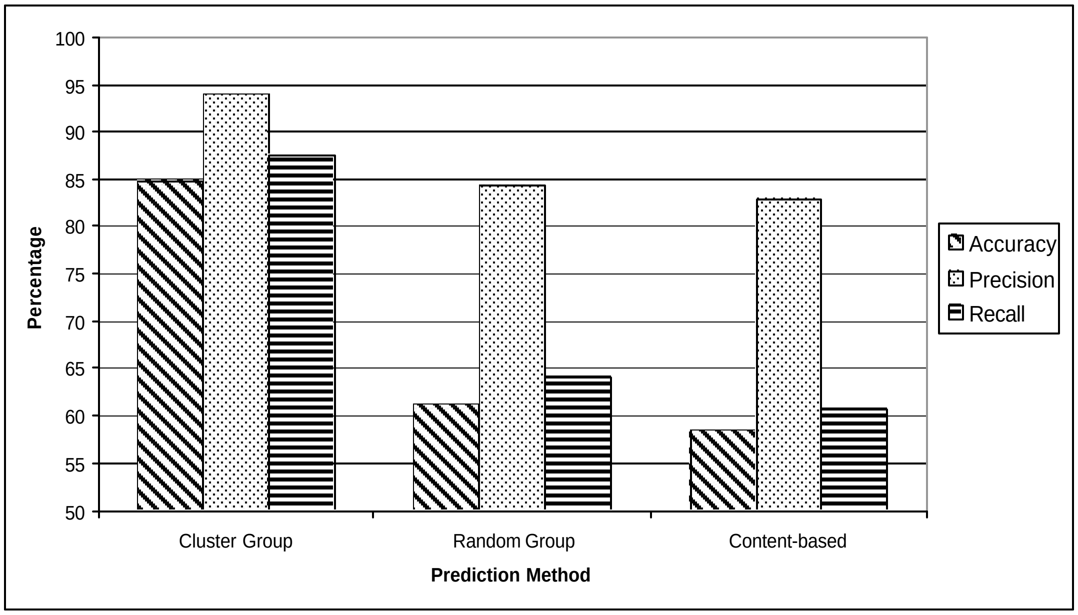

Figure 2: Comparison of algorithm results

5.0 Discussion

As was expected, the method of clustering users resulted in the greatest performance. Of the three, this method produced the highest levels of accuracy, precision and recall. However, a large amount of pre-processing of the ratings data is required to generate the clusters of users. For this reason, this method may not be appropriate for all applications.

While simpler to implement, the other two methods each produced reasonably high levels of accuracy, precision and recall to make them viable choices. Very little pre-processing is required for either of these methods, and their run-time requirements are much more in line with a real-time system. No matter which of the three methods is most appropriate for a specific application, each of them overcomes the problem of system performance.

This content-based approach to collaborative filtering is successfully able to overcome the difficulties outlined in section 1.0. A user that has rated only a few items is able to receive recommendations based on their feature list. A user must only rate one or two items in order for them to have a usable feature list. Additionally, it is now possible to give recommendations for an item that has only a few ratings. Instead of making predictions based on the ratings that an item has already received, the prediction is made by comparing the feature lists of the user directly to the feature list of the item. Finally, it is possible to compare users that have no common ratings, allowing the system to discover meaningful relationships between users that would not have otherwise been known. This also allows all users to be considered as potential neighbors, thus increasing the chances of finding similar users.

6.0 Future Work

There is still a great deal of work that can be done to improve upon this content-based approach to collaborative filtering. As mentioned in section 3.1, the efficiency of the clustering algorithm used by this study could be greatly improved. There are a number of efficient clustering algorithms that could be used instead, such as k-means or hierarchical clustering. Additionally, a more complex hybrid of collaborative filtering and content-based analysis could be developed to improve the predictions. A method proposed by [3] produces pseudo ratings by purely content-based means to create a dense ratings matrix. It then uses the combination of actual ratings and pseudo ratings to compare users. This approach eliminates the need for users to have co-rated items to be considered neighbors.

There exist many appropriate applications of recommender systems; this study has demonstrated that a content-based approach to collaborative filtering is a superior method for delivering such recommendations.

7.0 References

[1] *EachMovie Project*. http://research.compaq.com/src/eachmovie

[2] *Internet Movie Database (IMDb).* http://www.imdb.com

[3] P. Melville, R.J. Mooney and R. Nagarajan. "Content-boosted collaborative filtering." In *Proceedings of the SIGIR Workshop on Recommender Systems*, 2001.

[4] M.J. Pazzani. "A framework for collaborative, content-based and demographic filtering." *Artificial Intelligence Review*, 13(5-6):393-408, 1999.

[5] M. O'Connor and J. Herlocker. "Clustering items for collaborative filtering." *Presented at the Recommender Systems Workshop at Conference on Research and Development in Information Retrieval*, 1999.

[6] L.H. Ungar and D.P. Foster. "Clustering methods for collaborative filtering." Retrieved 2/7/2004 from http://www.cis.upenn.edu/~ungar/papers/clust.ps.

[7] "Computing Pearson's correlation coefficient." http://davidmlane.com/hyperstat/A51911.html

[8] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes and M. Sartin. "Combining content-based and collaborative filters in an online newspaper." *Presented at the ACM SIGIR Workshop on Recommender Systems*, Berkeley, CA, 1999.

[9] *MovieLens Project*. http://movielens.umn.edu