A simple hybrid movie recommender system

Jaldert Rombouts (rombouts@ai.rug.nl)

Tessa Verhoef (tverhoef@ai.rug.nl)

Abstract

A simple hybrid movie recommender system is described that combines content based and collaborative modelling and provides an explanation for increased user acceptance. The system uses rating data from the Netflix database which is linked to content information from the Internet Movie Database. In order to provide the user with insight into the reasoning behind a recommendation, the system creates an HTML page with a detailed explanation. Performance results are presented as well as a discussion on future improvements.

Keywords: Recommender systems; Collaborative filtering; Content based filtering; user acceptance; naive Bayes

Introduction

We live in an information society in which people are often confronted with very large amounts of data, for instance through the internet. We are asked to make choices that are almost impossible to make without additional information or guidance. Recommender systems can provide such guidance by assisting the user in the decision making process or by making the decision for the user. These systems use the enormous amount of available data in a way that users never can.

Recommender system are already being used in a lot of different domains. GroupLens (Resnick, Iacovou, Suchak, Bergstorm, & Riedl, 1994), for instance, is a system that recommends news articles that could be of interest to the user. AbeBooks recommends books, Last.fm helps the user to find new music and MovieLens, IMDb and Netflix recommend movies. Amazon.com is a recommender that is specialized in all three previously mentioned media.

Collaborative filtering

Varying approaches for automatic recommendation have been proposed. The oldest and most developed method is *collaborative filtering*. This method uses inter-user comparisons to generate new recommendations. A collaborative system consists of a database which contains the users' ratings and is augmented as the user interacts with the system over time. Users are compared based on their ratings and the obtained similarities and differences are used to make a recommendation. Collaborative filtering suffers from the sparsity problem. Not all users exploit the option to rate items they have seen or used. The available rating data is therefore typically very sparse, especially when a user is new or when the system is new and people are just starting to use it. Another problem is the first-rater problem: before an item has been recommended for the first time, the system will not recommend it. This problem applies to new items and obscure items which makes it less attractive for people with non-mainstream tastes. A virtue of collaborative filtering is that it can surprise the user with relevant items that are not explicitly similar to items in the users' profile. This so called 'outside the box' recommendation ability (Burke, 2002) is possible because it uses people-to-people correlations.

Content based methods

Another method uses *content based* information. Content based filtering uses item-to-item correlation to compare representations of content in an item to representations of content in items the user has rated. The similarities between items and the rating information are used to predict how much the user will like or dislike a new item. A disadvantage of this method is that it is completely dependent upon machine readable representations of items, which may be difficult to obtain. Content based methods are not able to surprise the user, because it uses the feature values of the items that the user has rated and will not recommend an item that does not share any of these values. Just as collaborative filtering, content based methods also suffer from the sparsity problem.

Hybrid methods

Collaborative filtering and content based methods have varying advantages and disadvantages. A combination of the two methods therefore provides a promising extension. Different ways have been proposed and used to combine two different recommenders (Burke, 2002). Weighting is a simple method that computes a recommendation from the results of all individual recommenders, for instance by linearly combining them or by a voting mechanism (Pazzani, 1999). This is a very straightforward method that makes easy adjustments possible, but it also uses the implicit assumption that the relative value of the different recommenders is uniform for all items, which is often not the case. As an alternative, switching techniques can be used, that use a criterion function to decide when to switch from one recommender to the other (Tran & Cohen, 2000). Basu, Hirsh, and Cohen (1998) proposed a method in which they use the information of one recommender as features for the other. The rating information from the collaborative part is used as an additional feature and content based filtering is done on the complete data set, including this new feature. Cascading different recommenders is another way to combine them which involves a staged process. Melville, Mooney, and Nagarajan (2002) for instance use a content boosted collaborative filtering technique, in which they first solve the sparsity problem by making virtual data using content based information, and then use collaborative filtering on this much larger data set.

Explanation

Recommender systems are widely used, especially in the online community, but up to now the application areas of the

technique have been limited to harmless decision making in the entertainment domain. For more serious issues, like booking a vacation, buying insurance or stock market decisions, people do not trust the technique enough to let it make these decisions for them. One way to overcome this distrust is to provide the user with an explanation (Herlocker, Konstan, & Riedl, 2000). In the eyes of a user, a recommender system is a black box, which makes it hard to understand why a certain decision has been made. An explanation can provide more insight into the reasoning behind the decision and also gives the user the opportunity to judge, according to this reasoning, whether to trust the decision or not. It should make the system more transparent. It has been shown experimentally that explanations increase the acceptance of both expert systems and recommender systems, or in general, decision support systems (Herlocker et al., 2000; Ye & Johnson, 1995).

In the following sections our hybrid movie recommender system is described. Collaborative filtering is linearly combined with a naive Bayesian content based approach to predict the movie preference of a user. The system provides the user with an insightful explanation that justifies the decision. First the problem domain and the system are described, then the result of experiments on the performance are reported and a discussion follows after that.

Method

Domain

The proposed system for movie recommendation uses both collaborative filtering and content based modelling and is supposed to perform the following task: for a certain user, decide which one of two proposed movies will probably be favored by the user. Two databases are combined and used: Netflix[1], providing rating information for the collaborative part, and the Internet Movie Database (IMDb)[2] for extracting content information. The Netflix database consists of rating information of 480189 different users. The ratings are ranged from one to five stars. One star indicates that a user does not like the movie and five stars means the user loves it. The database contains 17770 different movies, of which 8637 are used because these movies could be linked to the content information that is available from the Internet Movie Database (IMDb). To be able to use the information from both databases, a MySQL database was created from which the information could be extracted using simple queries. Because of the intense data-dependency and large calculations involved in collaborative filtering, we decided to use `PyFlix`[3], an off the shelf `Python` tool for accessing the Netflix dataset, which was a lot faster than our MySQL database.

[1] http://www.netflix.com/

[2] http://www.imdb.com/

[3] http://pyflix.python-hosting.com/

System

The system is a linear combination (50-50) of the predictions of the content based model and the collaborative filter discussed below.

Content based modelling The content based part of the movie recommender is based on a naive Bayesian text classification method (Mitchell, 1997). The classifier creates a naive Bayesian model for every user, based on the content of the movies the user has rated. The content that is used are the keywords, genres and actors of a movie and these features are assigned to an appropriate class: 1, 2, 3, 4 or 5, based on the rating for that movie. For every feature type, a separate model is created and the predictions of these models are linearly combined into one prediction. The number of possible feature values of the keywords and actors would be very large if all possible values were to be used since there is a huge amount of different keywords and different actors. To be able to keep the feature vectors manageable, only the keywords are used that occur more than 20 times (8762 in total) and only the actors are used that occur more than 50 times in the data set (34956 in total).

We are interested in the posterior probability of a certain rating/class (c), given the observation of movie features (o) for a new movie. Using Bayes Theorem, this can be defined as:

$$p(c|o) = \frac{p(c)p(o|c)}{p(o)} \tag{1}$$

Since the denominator in this equation does not depend on c and we can make use of the naive assumption that every feature f_i in the observation is conditionally independent of every other feature, we can rewrite the function as:

$$p(c|o) = p(c)\prod_{i=1}^{n} p(f_i|c) \tag{2}$$

where n is the number of features. In order to classify the movie, we need to find the maximum posterior probability of the five classes:

$$classify(f_i...f_n) = \max p(C=c)\prod_{i=1}^{n} p(F_i=f_i|C=c) \tag{3}$$

So, the class/rating with the highest posterior probability for this movie is the predicted rating on which the system bases its recommendation.

A disadvantage of this method is that it has to estimate the prior and conditional probabilities from the data, while the data is typically sparse. Direct probability calculation can therefore often give probabilities that are zero, which is undesirable. For this reason we use Laplacian smoothing (Mitchell, 1997) to eliminate zeros from the estimated probabilities. What this comes down to is that it adds a number of additional "hallucinated" examples. These examples are spread evenly over the possible values. In this case, every possible observation starts from a frequency of one, instead of

zero and the prior frequency of a class starts from the number of possible feature values, for instance the number of possible genres. Our estimate for the prior probability of a class now becomes:

$$p(c) = \frac{n_c + n_f}{n_t} \tag{4}$$

where n_c is the number of observations for class c, n_f is the number of possible feature values and n_t is the total number of observations. The estimate for the conditional probability becomes:

$$p(o|c) = \frac{n_o + 1}{n_c + n_f} \tag{5}$$

where n_o is the frequency with which a certain observation has been encountered in association with class c.

If the number of possible observations becomes very high, the probabilities can get very small, so in order to avoid underflow, the log-likelihoods are used.

The predictions of the three models are linearly combined into one recommendation in a way that yielded the best performance. It uses 60 percent of the prediction based on keywords, 30 percent of the prediction based on actors and 10 percent of the prediction based on genres. This is mostly due to the fact that actors and genres are less often known than keywords, and because there is much less variation in the genres.

Collaborative filtering The collaborative part of the system is heavily based on the system described by Herlocker, Konstan, and Riedl (1999), which is a neighbourhood-based approach to collaborative filtering.

The approach consists of three steps:

1. Calculate the similarity of the active[4] user to other users;
2. Select n users that are most similar to the active user: they form the neighbourhood;
3. Calculate a prediction based on a weighted combination of the neighbours ratings.

Like Herlocker et al. (1999), we use the Pearson correlation coefficient to calculate the similarity between two users:

$$P_{a,u} = \frac{\sum_{i=1}^{m}(r_{a,i} - \bar{r}_a) * (r_{u,i} - \bar{r}_u)}{\sqrt{\sum_{i=1}^{m}(r_{a,i} - \bar{r}_a)^2 * \sum_{i=1}^{m}(r_{a,u} - \bar{r}_u)^2}} \tag{6}$$

$P_{a,u}$ is the similarity between the active user a and another user u, $r_{i,j}$ is the rating that user i has given to item j, $\bar{r}_i$ is the average rating that user i has given, and m is the total number of items that both users have in common.

Given that the similarity only depends on the co-rated items of two users, it is possible to have a high correlation between users while this is in fact not the case. For instance, if two users have both rated 500 items, but have only two co-rated items, their correlation can still be perfect if they agree on these two items. In order to compensate for this effect, we adopted significance-weighting as proposed by Herlocker et al. (1999). This can be described by the following formula, where n is the number of overlapping items, and θ is the threshold value:

$$P_{a,u} = \begin{cases} n/\theta * P_{a,u} & \text{if } n < \theta; \\ P_{a,u} & \text{otherwise.} \end{cases} \tag{7}$$

We adopted a θ value of 50 which means that two users should have at least 50 overlapping items, or the corrrelation for this pair is reduced.

To calculate the similarities between all users $\frac{(n^2-n)}{2}$ comparisons are needed, where n is the total number of users. Given the large amount of users in the Netflix set, this is not feasible on a normal computer. In order to get the number of comparisons down, we did a preselection step on the users to evaluate as neighbours. First, only select users that have seen the item that we want to rate for the active user, and then randomly select 30000 of these users. So, in the worst case scenario, the algorithm needs to calculate 30000 similarities. We used this value because it yielded an acceptable runtime for evaluating one (user,item) pair on our hardware. Note that we only add users with a positive nonzero correlation to the list of possible neighbours: according to Shardanand and Maes (1995) including negative correlations makes no real difference.

The described strategy for selecting possible neighbours holds another advantage: we *know* that all neighbours have rated the item of interest, so all can have an influence on the final prediction. If you would just select the top-n users in the naive way, there is no guarantee that these neighbours have actually rated the item of interest, in which case their 'vote' would be lost.

The last step is combining the ratings of the neighbours into a prediction for the active user. The simplest approach would just take all ratings for the item by the neighbours, and multiply this by the weight of that neighbour, and divide the total by the total weight of the neighbours. Herlocker et al. (1999) show that this approach is not very good, because it does not compensate for the fact that every user rates differently. Some users only give five stars to items that they really love, and others are not as stringent with their ratings. A way to compensate for this is to use the deviation-from-mean average for all users, as detailed in Herlocker et al. (1999). In this way, when a user always gives high ratings to items, his vote will be worth less when his rating for the item of interest is also high.

The prediction algorithm consists of the average rating of the active user which is modulated by the ratings of the neighbours (our implementation uses 50 neighbours). The algorithm calculates its prediction as follows:

$$p_{a,i} = \bar{r}_a + \frac{\sum_{u=1}^{n}(r_{u,i} - \bar{r}_u) * P_{a,u}}{\sum_{u=1}^{n} P_{a,u}} \tag{8}$$

[4]This is the user for which we are calculating a prediction.

If the preselection step does not yield any neighbours for the user, the algorithm returns the best rating estimate it can find for that item. This estimate is the deviation-from-mean average for all users that have rated this movie plus the average of the active user. If the algorithm does find neighbours it calculates the the prediction as discussed.

Explanation To increase the user acceptance of the recommender system we created an explanation module that provides the user with detailed information in HTML format to justify the decision. The explanation gives the user insight into the reasoning behind the decision of both the collaborative model and the content based model and both algorithms also indicate how certain they are of this decision in order to give the user the opportunity to judge how trustworthy the decision is. The final decision is also presented accompanied by this certainty indication. There are three grades of certainty that the system can express: 'not very certain', 'reasonably certain' or 'certain'. The implementation for this is actually almost trivial: if the difference between predictions is small, it wil say that it is 'not very certain', and if the difference is almost a whole star, it will say that it is more confident. The thresholds for these decisions were found through manual experimentation. Figure 1 in appendix B gives an impression of the user interface.

The explanation of the content based model is provided to the user in a small story. This story tells the user which features of the two movies caused the system to recommend one movie over the other. For instance, because the actors in one movie have been associated with higher movie rankings than the actors in the other movie. The difficulty with these explanations is that it is very hard to determine which type of feature is responsible for a high or a low rating. The recommender only takes the keywords, actors and genres into account, but what a person likes or dislikes about a movie might just as well be caused by the influence of a director or a certain style of filming. Due to this 'credit assignment problem' the explanation is purely based on the conclusions drawn by the naive Bayesian model which does not always necessarily correspond with the real preferences of the user. The user will be able to recognize such a deviation from his or her own preferences and can decide whether to trust the recommendation or not. The credit assignment problem will be further discussed in the discussion section.

We have based the explanation of the collaborative part of our system on the work of Herlocker et al. (2000). Because the system is actually a complex mathematical model, spelling out how the algorithm works does not count as an explanation. They concluded that a simple graph showing what the neighbours had rated worked best. We implemented this feature for our explanation: it is included on the bottom right of the user interface, see figure 1 in appendix B. Because the system must give a ranking for two movies, neighbours for two movies are visible (if there are any). There are only three groups, where the middle group indicates the number of neighbours that have rated the movies with tree stars. If the distribution has its center of gravity to the right, the neighbours thought that the movie was good, and if it is at the left, they thought that it was a bad movie. Note that this is not the exact information that the system uses, since some users have more weight than others and these weights are combined as shown in equation (8), but in practice this graph gives a clear impression, which almost always corresponds to the decision. You can also see on how much neighbours the decision is based, which is also some indication for the correctness of the decision of the algorithm.

Performance

The performance of the hybrid recommender was tested, but also the performance of the individual recommenders. The task is defined as a ranking task in which the recommender has to decide which movie of two movies that are new for the user will be the most favorable. The predicted rankings are compared to the actual rankings with a repeated cross-validation approach. The ranking is based on a comparison of the two predicted ratings for the movies. To provide more insight into the performance for these predictions, these estimated ratings are also tested. A Root Mean Squared Error calculation is used to determine the error in the prediction (p) compared to the real rating (r) after n predictions:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(p-r)^2} \tag{9}$$

The performance of the individual recommenders was tested using cross-validation. In ten trials, 100 random users (who had rated at least two movies for which content information was available) were selected. Two movies were extracted from the total amount the user had rated and the models were created using the data of the remaining movies. Then the recommenders were asked to classify the two selected movies and the predicted rankings and ratings were compared.

Determining whether the ranking is correct is not as trivial as it seems: for instance, what is the correct answer when the user has given both test items the same rating? We decided that any ranking would count as correct in this case.

The result of the ten trials shows that the average number of correct rankings for the content based model is 86% with a standard deviation of 3.37%. The ranking performance of the collaborative model is 88% with a standard deviation of 2.65%. The predicted ratings of the content based model deviated from the actual ratings with an average RMSE of 0.87 with a standard deviation of 0.04. The rating error of the collaborative model was on average 0.88 with a standard deviation of 0.03.

In our implementation, the combined classifier was much slower than the individual classifiers, taking about 1-4 minutes per test case. This is the reason why we did not test the combined classifier as extensively as the individual classifiers. The result was a correct ranking of 88.25%, with a standard deviation of 1.5%. The average RMSE was 0.83

with a standard deviation of 0.05 This is based on four experiments with 50 tests each, so it is hard to say how this compares to the individual components.

Discussion

It is very hard to compare our results to other work, because almost all writers use different databases for training and validating their algorithm. Even if the writers have used the same database it can still be that the results are hard to compare because the authors have made different assumptions and their algorithms have different preconditions. In our case for instance, both test movies need to be linked to the IMDB database in order to get a prediction. So, users have seen at least two movies, but given that the probability that we have linked a movie with IMDB is about 50% you would expect that these users have seen more than two movies. In this way, our algorithm sidesteps one of the real problems of recommender systems: doing predictions for users that have seen very few items. It should be clear that our system is not nearly as good as that of the current leaders in the Netflix-challenge[5], which have an RMSE of about 0.86!

Both our systems, and most recommender systems in general, are highly dependent on ratings that users have given for certain items. It is an important question whether these ratings can be considered reliable. The five point rating scale that is used by Netflix is an arbitrary scale and it has been shown that changing the scale's range can alter the distribution of responses (Amoo & Friedman, 2001). Excluding a middle choice, for instance, has been found to bias towards positive responses. The rating behaviour of users is also influenced by the predictions that recommender systems make (Cosley, Lam, Albert, Konstan, & Riedl, 2003), which makes it harder for the system to accurately model the users' preferences. To improve the performance of recommender systems in the future, it might be beneficial to consider alternative rating mechanisms.

The credit assignment problem that was mentioned earlier is a problem that not only affects the explanation that the system gives, but it is a general disadvantage for content based methods that we cannot objectively determine which part of the content was the deciding factor behind a high or a low rating. Why do people like 'The Godfather' so much? Is it the actors, or the genre? Or do people like it because it is directed by Francis Ford Coppola and they like his style? We would never no unless we ask the user. Predictions of content based recommenders could improve if we would ask the user why they give a specific rating.

There are a few disadvantages of the neighbourhood preselection as used in the collaborative part of the system. The first is perhaps conceptual: the neighbours for both movies are not actually the *same* neighbours, there does not have to be any overlap between these different groups of users, while the interface gives the impression that this is the case. Another problem is that the system does not have to be consistent in a rating for a user and the same two items, because it selects random users in the preselection stage and if there are more than 30000 users that have seen a items, there is a probability that the top-50 neighbours will not be the same. This is a trade-off that we needed to make in order to achieve a feasible runtime. Whether the predictions provided by our algorithm are worse than neighbours calculated in the naive way is something that is worth looking into.

A point for improvement lies in the way that the collaborative system calculates the correlations between two users. The solution offered by Herlocker et al. (1999) that we adopted seems more like a hack than a real solution for this problem. There probably are statistical tools that will offer a more solid foundation for this decision, e.g. where the confidence interval becomes larger if the overlap between two users becomes smaller. This will almost certainly have a positive effect on the predictions.

Another point for improvement is the way the final predictions are calculated: if a user has a large number of neighbours for the first item and a very small number of neighbours for the second item, the votes of the latter neighbours will weigh more heavily. This is also the case when you use normal neighbourhood-based methods, in which not all neighbours have rated both items.

The combination of the classifiers was done in a linear fashion, because this is relatively easy to do by hand. But, it is probably the case that the weights that we used for combining them are probably not the best ones. Also, it is possible that a linear combination of the classifiers not the best solution to combine them.

It also needs to be mentioned that we did not evaluate the user-interface of the recommender system: we followed the suggestions of Herlocker et al. (2000) and our own intuitions about what kind of explanations users would find acceptable, but we did not test the system on real end-users. This is something that needs to be done in future research.

In conclusion, it has been shown that a simple hybrid recommender system performs reasonably well on a movie recommendation task and can also provide the user with an insightful explanation that makes the system and its separate recommenders more transparent. The advantage of using a combination of different classifiers is not only that they compensate for each others shortcomings, but also that the combination of explanations provides a detailed description of the reasoning behind the recommendation.

References

Amoo, T., & Friedman, H. (2001, February). Do numeric values influence subjects' responses to rating scales? *Journal of International Marketing and Marketing Research*, *26*, 41-46.

Basu, C., Hirsh, H., & Cohen, W. (1998). Recommendation as classification: Using social and content-based information in recommendation. *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, *714720*.

[5] http://www.netflixprize.com/leaderboard

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

Cosley, D., Lam, S., Albert, I., Konstan, J., & Riedl, J. (2003). Is seeing believing?: how recommender system interfaces affect users' opinions. *Proceedings of the SIGCHI conference on Human factors in computing systems*, 585–592.

Herlocker, J., Konstan, J., & Riedl, J. (1999). An algorithmic framework for performing collaborative filtering. *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, 230–237.

Herlocker, J., Konstan, J., & Riedl, J. (2000). Explaining collaborative filtering recommendations. *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, 241–250.

Melville, P., Mooney, R., & Nagarajan, R. (2002). Content-Boosted Collaborative Filtering for Improved Recommendations. *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, 187–192.

Mitchell, C. (1997). *Machine Learning*. The McGraw-Hill Companies, Inc.

Pazzani, M. (1999). A Framework for Collaborative, Content-Based and Demographic Filtering. *Artificial Intelligence Review*, *13*(5), 393–408.

Resnick, P., Iacovou, N., Suchak, M., Bergstorm, P., & Riedl, J. (1994). GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work* (pp. 175–186). ACM.

Shardanand, U., & Maes, P. (1995). Social information filtering: algorithms for automating "word of mouth". *Proceedings of the SIGCHI conference on Human factors in computing systems*, 210–217.

Tran, T., & Cohen, R. (2000). Hybrid Recommender Systems for Electronic Commerce. *Knowledge-Based Electronic Markets, Papers from the AAAI Workshop, Technical Report WS-00*, *4*.

Ye, L., & Johnson, P. (1995). The Impact of Explanation Facilities on User Acceptance of Expert Systems Advice. *MIS Quarterly*, *19*(2), 157–172.

Appendix

A. Item-item collaborative filtering

Another collaborative modelling algorithm was developed which uses *item-item* similarity, based on comparing user ratings for these items. The algorithm is very simple:

1. Calculate the distance between the movie of interest and all movies that the active user has seen;
2. Normalize the distances, so the largest distance equals 1
3. Calculate the prediction based on these distances.

We use the Euclidean distance as a distance measure:

$$D(a,b) = \sqrt{\sum_{i=1}^{m} (r_{i,a} - r_{i,b})^2} \tag{10}$$

Where m is the number of users that two movies have in common and $r_{i,x}$ is the rating for an item x of such a user i. Note that this approach suffers from the same problem as the other algorithm: if the overlap is small, the calculated distance will not be a realistic estimate. We adapted the significance weighing strategy as follows to work with distances instead of correlation values ($\theta = 10000$):

$$D(a,b) = \begin{cases} D(a,b) * (\theta/n) & \text{if } n < \theta; \\ D(a,b) & \text{otherwise.} \end{cases} \tag{11}$$

Now, all distances are normalized by dividing all distances by the largest distance found, so every distance is a value in the interval $[0,1]$.

In order to calculate a prediction we select the top-15 items that are most similar to the item of interest. Now we use a simple calculation for the prediction:

$$p_{a,i} = \sum_{j=1}^{15} (1 - D(i,j)) * r_{a,j} \tag{12}$$

Where $p_{a,i}$ is the prediction for the rating that user a will give the item of interest i, and $r_{a,i}$ is the rating that user a has given item j.

The performance of this algorithm was not quite as good as we expected, often scoring worse than a simple model that always returned a rating of 3 (average rmse was 1.55 (standard deviation 0.10) for 5 groups consisting of 100 tests). We thought it was nice to include because it is a collaborative approach that works on different data than the user-user based approach, and perhaps it will thus prove useful when combining classifiers.

B. Screenshot of the user interface

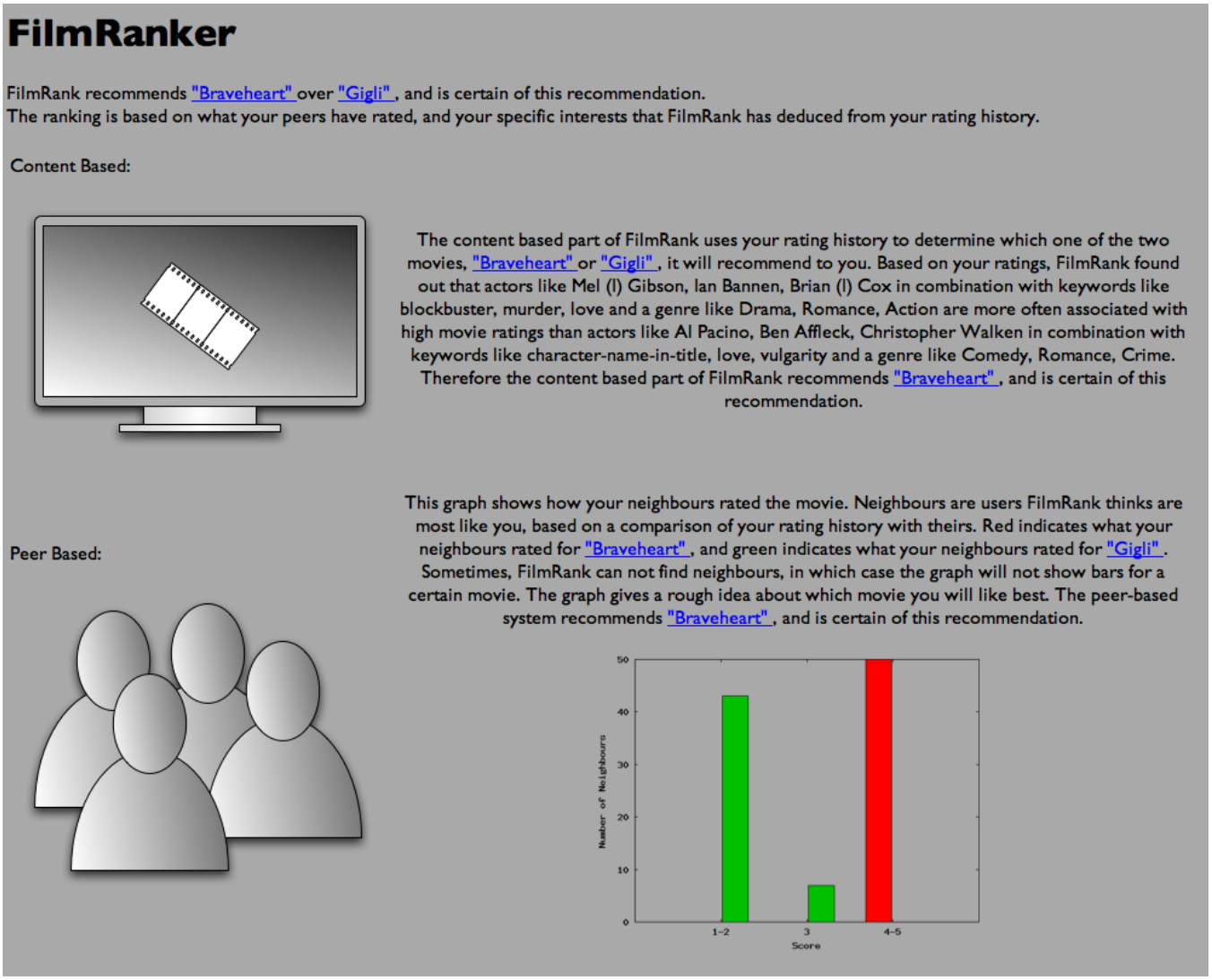

Figure 1: Screenshot of the user interface. The result of the combined classifier is on the top, and the explanations for the content-based part and the collaborative part are indicated as 'Content Based' and 'Peer Based' respectively. For reference, this prediction is for user 7 in the netflix dataset, and compares 'Gigli' to 'Braveheart'.