# Collaborative Filtering with the Simple Bayesian Classifier

Koji Miyahara[1] and Michael J. Pazzani[2]

[1] Information Technology R&D Center
Mitsubishi Electric Corporation
5-1-1 Ofuna, Kamakura, Kanagawa 247-8501, JAPAN
miya@isl.melco.co.jp
[2] Department of Information and Computer Science
University of California, Irvine
Irvine, CA 92697-3425, USA
pazzani@ics.uci.edu

**Abstract:**
*Many collaborative filtering enabled Web sites that recommend books, CDs, movies, videos and so on, have become very popular on Internet. They recommend items to a user based on the opinions of other users with similar tastes. In this paper, we discuss an approach to collaborative filtering based on the simple Bayesian classifier. The simple Bayesian classifier is one of the most successful supervised machine-learning algorithms. It performs well in various classification tasks in spite of its simplicity. In this paper, we define two variants of the recommendation problem for the simple Bayesian classifier. In our approach, we calculate the similarity between users from negative ratings and positive ratings separately. We evaluated these algorithms using a database of movie recommendations and joke recommendations. Our empirical results show that one of our proposed Bayesian approaches significantly outperforms a correlation-based collaborative filtering algorithm. The other model almost outperforms as well although it shows similar performance to the correlation-based approach in some parts of our experiments.*

**Keywords:**
Agents, User modeling

**Email address of contact author:**
**miyahara@ics.uci.edu** (Until 3/31/00)
**miya@isl.melco.co.jp** (After 4/1/00)

**Phone number of contact author:**
+1- (949) 824-8046 (Until 3/31/00)
+81-(467) 41-2486 (After 4/1/00)

# 1 Introduction

The growth of Internet has resulted in a tremendous amount of information available and a vast array of choices for consumers. Recommender systems are designed to help a user cope with this situation by selecting a small number of options to present the user. They filter and recommend items based on a user's preference model. Various types of recommender systems have been proposed so far, their filtering techniques fall into two categories. One is content-based filtering (e.g. [12]) and the other is collaborative filtering or social filtering (e.g. [16]).

In content-based filtering, a user's preference model is constructed for the individual based upon the user's ratings and descriptions (usually, textual expression) of the rated items. Such systems try to find regularities in the descriptions that can be used to distinguish highly rated items from others. On the other hand, collaborative filtering tries to find desired items based on the preference of set of similar users. In order to find out like-minded users, it compares other users' ratings with the target user's ratings. It is not necessary to analyze the contents of items, therefore it can be applied to many kind of domains where a textual description is not available or regularities in the words used in the textual description are not informative (e.g. [4]). One of the most popular algorithms in collaborative filtering is a correlation-based approach. In this paper, we report experimental results comparing the collaborating filtering with the Simple Bayesian Classifier as an alternative approach.

This paper is organized as follows. We present the central ideas of current typical collaborative filtering algorithms. We define the two alternative formulations of the Simple Bayesian Classifier for collaborative filtering. Then, we evaluate our algorithms on database of user ratings for movies and jokes, and show that our approach outperforms the correlation-based collaborative filtering algorithm. Finally, we discuss the results and summarize this paper.

# 2 Collaborative Filtering

The main idea of collaborative filtering is to recommend new items of interest for a particular user based on other users' opinions. A variety of collaborative filtering algorithms have been reported and their performance has been evaluated empirically ([2], [15], [16]). These algorithms are based on a simple intuition: predictions for a user should be based on the preference patterns of other people who have similar interests. Therefore, the first step of these algorithms is to find similarities between user ratings. Resnick et al. [15] use the *Pearson correlation coefficient* as a measure of preference similarity. The correlation between user *j* and *j'* is:

$$w_{jj'} = \frac{\sum_i (R_{ij} - \overline{R}_j)(R_{ij'} - \overline{R_{j'}})}{\sqrt{\sum_i (R_{ij} - \overline{R}_j)^2 \sum_i (R_{ij'} - \overline{R_{j'}})^2}}$$

where all summations over *i* are over the items which have been rated by both *j* and *j'*. The predicted value of user *j* for item *i* is computed as a weighted sum of other users' ratings:

$$\hat{R}_{ij} = \overline{R_j} + \frac{\sum_{j'} (R_{ij'} - \overline{R_{j'}}) w_{jj'}}{\sum_{j'} \left| w_{jj'} \right|}$$

These correlation-based prediction schemes were shown to perform well. However, it should be valuable to think of other approaches. Breese et al. [2] report a variety of modification to the above typical collaborative filtering techniques and the use of Bayesian clustering and a Bayesian network. A primary difference between what we propose below and the work of Breese et al. [2] is that we construct a separate Bayesian model for each user. This is practical only for the simple Bayesian classifier that is linear in the number of examples and number of features.

## 3 Simple Bayesian Model

### 3.1 Rating Matrix

Most of collaborating filtering systems adopt numerical ratings and try to predict the exact numerical ratings. However, we are not interested in the prediction of the exact rating a user would have given to a target item. We would much rather like to have a system that can accurately distinguish between items to recommend and others. Therefore, we defined two classes, *like* and *dislike*, that were used as class labels. Table 1 is an example of rating matrix in which three users have reported ratings on five different items. Some entries in the matrix are empty because users do not rate every item. The last row represents the ratings of a user for which the system will make a prediction. Typically, the rating matrix is *sparse* because most users do not rate most items.

| | **$I_1$** | **$I_2$** | **$I_3$** | **$I_4$** | **$I_5$** |
|---|---|---|---|---|---|
| **$U_1$** | *Like* | *Dislike* | *Dislike* | | *Like* |
| **$U_2$** | *Dislike* | | | *Dislike* | *Dislike* |
| **$U_3$** | | *Like* | *Like* | | *Like* |
| **Class Label** | *Like* | *Dislike* | *Like* | *Like* | *?* |

**Table 1: Example of User Ratings in a sparse matrix**

| | **$I_1$** | **$I_2$** | **$I_3$** | **$I_4$** | **$I_5$** |
|---|---|---|---|---|---|
| **$U_1$like** | *1* | *0* | *0* | *0* | *1* |
| **$U_1$dislike** | *0* | *1* | *1* | *0* | *0* |
| **$U_2$like** | *0* | *0* | *0* | *0* | *0* |
| **$U_2$dislike** | *1* | *0* | *0* | *1* | *1* |
| **$U_3$like** | *0* | *1* | *1* | *0* | *1* |
| **$U_3$disike** | *0* | *0* | *0* | *0* | *0* |
| **Class Label** | *Like* | *Dislike* | *Like* | *Like* | *?* |

**Table 2. Boolean Feature transformation of Ratings Matrix**

Billsus and Pazzani [1] proposed transforming the format of rating matrix so that every cell has an entry. In their format, each user's ratings are divided into two features, which has a Boolean value indicating whether the user reported liking the item and whether the user reported not liking the item. Table 2 shows the resulting Boolean feature based rating matrix. One of the advantages of this transformation is that we can treat each feature as an attribute like a word in text-based information filtering domains. Therefore. it is possible to apply any supervised machine learning algorithm to the collaborative filtering task.

### 3.2 Simple Bayesian Classifier

The Simple Bayesian Classifier is one of the most successful algorithms on many classification domains. Despite of its simplicity, it is shown to be competitive with other complex approaches especially in text categorization and content based filtering. Making the "naïve" assumption that features are independent given the class label, the probability of an item belonging to class *j* given its *n* feature values, $p(class_j| f_1, f_2, \ldots f_n)$ is proportional to:

$$p(class_j)\prod_i^n p(f_i \mid class_j)$$

where both $p(class_j)$ and $p(f_i \mid class_j)$ can be estimated from training data. To determine the most likely class of an example, the probability of each class is computed, and the example is assigned to the class with the highest probability. Although the assumption that features are independent given class label of an item is not realistic in this domain, the Simple Bayesian Classifier has been shown to be optimal in many situations where this assumptions does not hold [3] and has been empirically been shown to be competitive with more complex approaches in many others (e.g., [9]).

Here, we define two variants of the Simple Bayesian Classifier for collaborative filtering.

(1) *Transformed Data Model*

This model is identical to the multi-variate Bernoulli model applied to the transformed data such as that in Table 2. This model assumes that all the features, even dual features ($U_i like$ and $U_i dislike$), are completely independent. After selecting a certain number of features, absent or present information of the selected features is used for predictions. That is:

$$p(class_j| f_1=1,\ f_2=0\ , f_3=1 \ldots f_{n-1}=1, f_n=0)$$

where $f_i=1$ means that $f_i$ is present on the target item and $f_i=0$ means that $f_i$ is absent on the target item.

When estimating conditional probabilities, e.g. $p(f_i\ =1|\ class_j)$ , it is calculated over all ratings of the target user. The following conditions hold for this model:

$$p(U_i like = 1 \mid class_j ) + p(U_i like = 0 \mid class_j ) = 1.$$
$$p(U_i dislike = 1 \mid class_j ) + p(U_i dislike = 0 \mid class_j ) = 1.$$

However, $p(U_i like = 1 \mid class_j ) + p(U_i dislike = 1 \mid class_j )$ and $p(U_i like = 0 \mid class_j ) + p(U_i dislike = 0 \mid class_j )$ does not necessarily equal 1 because some users may have not indicated whether they like or dislike a particular item.

(2) *Sparse Data Model*

> In this model, it is assumed that only known features are informative for classification. Therefore, only known features are used for predictions. Therefore the following formula is considered as follows:
>
> $$p(class_j |\ f_1=1,\ f_3=1, \ldots f_{n-1}=1)$$
>
> Moreover, we make an only use of the data which both users in common rated when estimating conditional probabilities. In this representation, the following condition holds:
>
> $$p(U_i like = 1 \mid class_j ) + p(U_i dislike = 1 \mid class_j ) =1$$

For example, in the rating matrix of Table 2, the estimated conditional probability of $p(U_1 like \mid like) = 0.33$ in the transformed data model and $p(U_1 like \mid like) = 0.5$ in the sparse data model[1] respectively.

By using Simple Bayesian Classifier to make predictions, we expect to avoid a problem with correlation-based collaborative filtering algorithms. The correlation-based algorithms make a global similarity model between users, rather than separate models for classes of ratings (e.g. positive rating vs. negative rating). It might be possible that a set of one user's positive ratings is a good predictor for other users' positive ratings but the negative ratings of one user may not be useful in making predictions. Since the proposed models treat each class of ratings separately, we expect that the Bayesian model will capture similarity between users more precisely.

### 3.3 Feature Selection

Feature selection is a common preprocessing technique in many supervised learning algorithms. By restricting the number of features, it might be expected that it would increase the accuracy of the learner by ignoring irrelevant features or reduce the computation time. We apply a feature selection method to find a set of the *n* most informative features. Since our goal is to discriminate between classes, we define *informative* as being equivalent with *providing the most information* about an item's class membership. Intuitively, we would like to select features that appear more frequently in one class than in others. We use an information theory based approach to determine *n* most informative features. This is accomplished by computing the expected information gain [13] that the presence or absence of a feature *F* gives toward the classification of a set of labeled items *S*:

$$E(F, S) = I(S) - [p(F = 1)\ I(S_{F = 1}) + p(F = 0)\ I(S_{F = 0})]$$

*where* $p(F =1)$ is the probability that feature *F* is present on an item, and $S_{F = 1}$ is the set of items for which $F = 1$ holds, and $I(x)$ is the entropy of a set of labeled items, defined as:

[1] We use Laplacean prior in the actual calculation of conditional probabilities to smooth the probability estimates with few rating and to avoid estimating a probability to be 0. Therefore, the values of $p(U_1 like \mid like)$ are (1+1)/(3+2) = 0.4 in Transformed Data Model and (1+1)/(2+2) = 0.5 in Sparse Data Model respectively.

$$I(S) = \sum_{c \in classes} - p(S_c) \log_2 (p(S_c))$$

where $S_c$ is the set of all rated items that belong to class *c (In our case, c = {like, dislike})* by the target user. This formula is suitable for the *Transformed Model*, because it can calculate information gain of all the features independently even dual features such as $U_i$*like* and $U_i$*dislike*. However the above formula doesn't guarantee that selected feature is informative in the *Sparse Model*, because the conditional probability is estimated from the common items rated by both users. For *Sparse Model*, we extend the above formula as follows:

$$E_{sparse}(F, S) = E(F,S_{common})*E(F,S)*\ E(\emptyset\ F,S)$$

where $S_{common}$ is the set of the common rated items by both users and $\emptyset\ F$ is the dual feature of *F*, that is $\emptyset\ U_i like = U_i dislike$ and $\emptyset\ U_i dislike = U_i like$. Note that since $E(U_i like,\ S_{common})$ is equal to $E(U_i dislike,\ S_{common})$, the value of $E_{sparse}(U_i like,\ S)$ is equivalent to the value of $E_{sparse}(U_i dislike,\ S)$ in the above formula. Accordingly, selecting features is identical to selecting users in the *Sparse Model*.

# 4 Experiments

## 4.1 Dataset

We used the *EachMovie dataset* and *Jester dataset* as test data.

*(1) EachMovie dataset*

The *EachMovie* service was part of a research project at the DEC Systems Research Center [10]. The service was available for an 18-months period and was shut down in September 1997. During that time 72,916 users entered numeric ratings for 1,628 movies. User ratings were recorded on a numeric six-point scale, ranging from 0 to 1 (0, 0.2, 0.4, 0.6 0.8, 1.0). In our experiment, we use an experimental protocol similar to the one first used in [1]. We restricted the number of users to first 2,000 users in the database. These 2,000 users provided ratings for 1,410 different movies.

*(2) Jester Data*

*Jester* is a WWW-based joke recommendation system, which has been developing at University of California, Berkeley [4]. This data has 21,800 users entered numeric ratings for 100 jokes. User ratings were recorded on a real value, ranging from -10 to +10. Like EachMovie dataset, we restricted the number of users to first 3,000 users in the database. These 3000 users provided rating for 100 different jokes.

## 4.2 Evaluation Criteria

We are not interested in the most accurate prediction of the exact rating which a user would have given to the target item. Rather we would like to have a system that can accurately distinguish items that are liked by the user and items disliked. To distinguish items, we transformed numerical ratings into these two labels. We labeled items as *like* if the numerical rating for the item was above 0.7 (midpoint between the two possible user ratings *0.6* and *0.8*), or *dislike* otherwise in EachMovie dataset. And, we labeled items as like if the numerical rating for the item was above 2.0, or dislike otherwise in Jester dataset.

Not only does assigning class labels allow us to measure classification accuracy, we can also apply additional performance measures, *precision* and *recall,* commonly used for information retrieval tasks. However, it might be easy to optimize each of these measurements. To avoid this problem, we use F-Measure [7], which combines *precision* and *recall*:

$$F-Measure=\frac{2\cdot precision\cdot recall}{precison+recall}$$

### 4.3 Experimental Methodology

In our first experiment, we have evaluated the effectiveness of feature selection for our proposed two alternative representations. We also have evaluated a typical correlation-based approach, which is described in [15]. We selected 20 test users who have rated at least 80 movies from the EachMovie and 20 test users who have rated at least 60 jokes from the Jester dataset respectively. In the correlation-based approach, after calculating the predicted score, we labeled like or dislike according to the thresholds (0.7 in EachMovie and 2.0 in Jester). For each test user we ran a total of 20 paired trials for each algorithm, where we varied the number of features. For an each trial in an experiment, we randomly selected 50 rated items for EachMovie and 40 for Jester as a training set, and 30 items for EachMovie and 20 for Jester as a test set. The final results for one user are then averaged over all 20 trials, and we report the average value over 20 test users. In the *Transformed Model*, we selected N most informative features from all of the features. Predictions are made using all of these selected features. However, since *Sparse Model* uses only present features, it would happen that no selected feature is present for the target items. To avoid this problem, we picked up N most informative users (features) among users have already rated the target item. We applied similar method to the correlation-based approach, except it used an absolute value of correlation coefficient instead of the expected information gain.

In our second experiment, we have evaluated performance when we change the number of rated item in a training data. We started training with 10 rated items and increased the training set incrementally in steps of 10 up to 50 items for EachMovie and up to 40 items for Jester, measuring the algorithms' performance on the test set for each training set size. For each algorithm, we set the number of features, which got the best performance at the first experiment. Like the first experiment, for each test user, we ran a total of 20 paired trials for each algorithm. We repeated this for all test users and the final results reported here are averaged over 20 test users.

## 5 Results

Figure 1 shows the classification accuracy of our first experiment with EachMovie dataset. Note that since there are 4,000 (2000 users × 2 features per user) features at most in EachMovie dataset, maximum number of features is 3,998 (4,000 features – 2 features) in the *Transformed Model*. The results show that the *Transformed Model* reaches a maximum accuracy of 67.3% at 100 features. It seems to be sensitive to the number of selected features and it is getting worse in proportion to the number of features significantly. The *Sparse Model* reaches a maximum classification accuracy of 71.6% at 200 users. However, there are no significant differences among the performance with 30 user or more. *Correlation* reaches a maximum accuracy of 66.4% at 30 users. Its performance seems to be stable at 30 users or more like the *Sparse Model*.

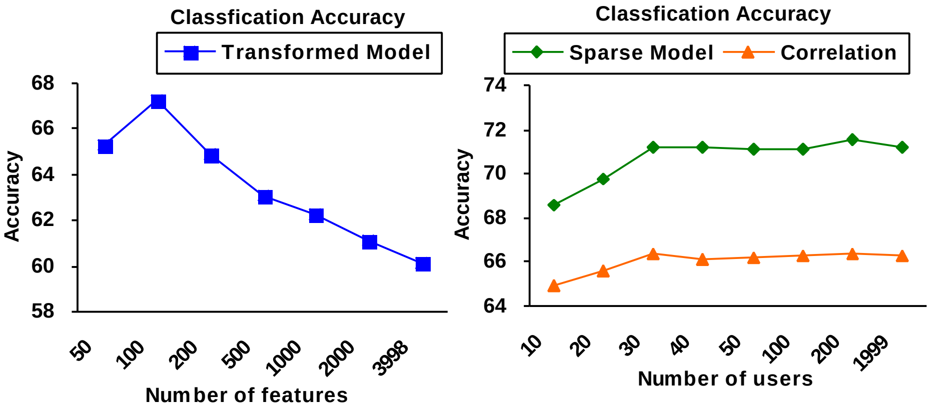

**Figure 1: Effects of Feature Selection (EachMovie Dataset)**

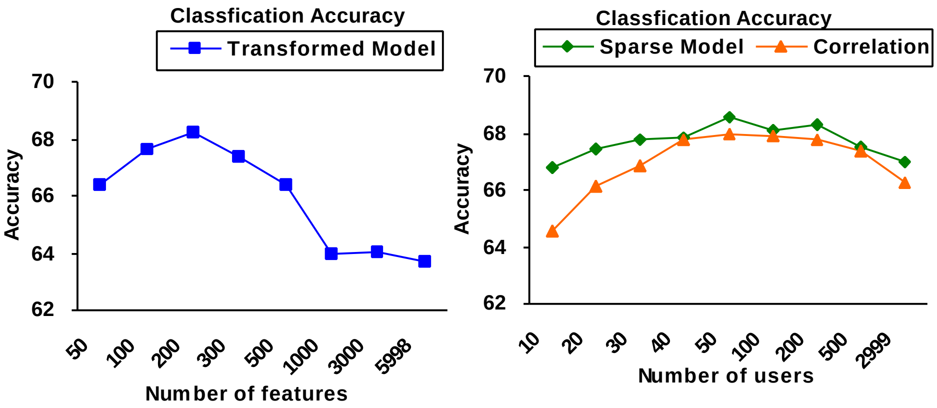

**Figure 2: Effects of Feature Selection (Jester Dataset)**

Figure 2 shows the classification accuracy with Jester dataset. The *Transformed Model* reaches a maximum classification accuracy of 68.3% at 200 features, *Sparse Model* is 68.5% at 50 users and *Correlation* is 68.0% at 50 users. Like EachMovie dataset, *Transformed Model* clearly has an optimal number of features. The performance of *Sparse Model* and *Correlation* is getting worse when picking larger number of users.

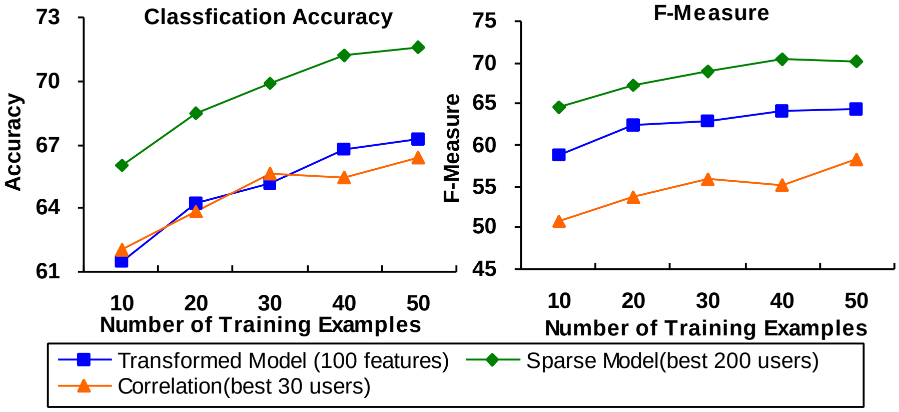

**Figure 3: Learning Curves (EachMovie Dataset)**

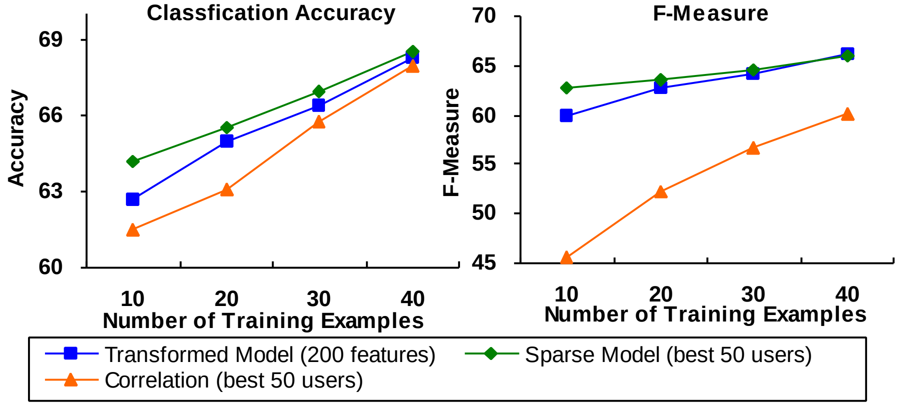

**Figure 4: Learning Curves (Jester Dataset)**

Figure 3 shows learning curves in our second experiment with EachMovie dataset, and Figure 4 shows learning curves with Jester dataset. For each model, we set the number of features which performs best in our first experiment. These results show that the *Sparse Model* performs best among three models. Especially, with EachMovie dataset, it significantly outperforms other two models significantly (as for accuracy, at 50 training examples, 71.6% for *Sparse Model* vs. 67.3% for *Transformed Model* and 66.4% for *Correlation*. As for F-Measure, 70.2% for *Sparse Model* vs. 64.4% *Transformed Model* and 58.4% for *Correlation*). *Transformed Model* generally performs better than *Correlation* except the accuracy at 30 or less training examples. With Jester dataset, *Sparse Model* performs slightly better than the *Transformed Model* and both models outperform *Correlation*. At 40 training examples, *Transformed Model* reaches F-Measure of 66.3%, *Sparse Model* is 65.9% and *Correlation* is 60.1%. The accuracy of *Sparse Model* is 64.2%, *Transformed Model* is 62.7% and

*Correlation* is 61.7% at 10 training examples. As for the F-Measure, *Sparse Model* is 62.8%, *Transformed Model* is 60.0%, and *Correlation* is 45.6%.

## 6 Discussion

Our experimental results show that our proposed collaborative filtering with the Simple Bayesian Model performs well. The *Sparse Model* significantly outperforms over the correlation-based algorithm. We think that the probability calculation by dividing positive ratings and negative ratings separately captures more precise similarity between users and it has a good effect on predictions. We also think that probability smoothing by Laplacean prior in the *Sparse Model* might be effective especially in the case that the number of commonly rated items is small.

The feature selection is effective in the *Transformed Model*, the model has an optimum number of features which can work best. Similar results are reported in text classification tasks using the multi-variate Bernoulli model of Simple Bayesian Classifier ([9], [12]). The *Sparse Model* has an optimum number of features with Jester dataset, but it is not significant. The effect of features selection is not clearly shown with EachMovie dataset.

It is interesting that the *Transformed Model* works well. This model treats even missing ratings as informative. It might be possible that this treatment distorts the similarity between users, because it would be happen that a user doesn't rate items by chance. In text classification tasks, it is reasonable that an absent word is informative, because people tend to use suitable words for its domain of the text. Therefore, absence of the word is a good predictor for its negative class. However, our empirical results show that this model performs well in spite of this intuitive thought. One advantage of the *Transformed Model* is that it greatly reduces computational complexity. Once the model selected fixed number of features, it can make predictions using the selected features.

## 7 Conclusions and Future Work

In this paper, we reported on collaborative filtering with the Simple Bayesian Classifier. We proposed two representations for the Simple Bayesian Classifier. We found that the Sparse Data Model performs better than the Transformed Data Model and the typical correlation-based approach. This shows that the transformation proposed by Billsus and Pazzani [1] to use any machine learning algorithms for collaborative filtering may be improved upon by algorithms which handle missing data well. The Transformed Data Model also outperforms the correlation-based approach although it shows similar accuracy to the correlation approach in some parts of the experiment with EachMovie dataset. Since our experiments used two datasets, it is important to adopt other type of dataset to verify our methodology.

In future work, we will investigate to combine content based filtering and collaborative filtering. As a first step, we plan to integrate keyword features and user features within one framework using the Simple Bayesian Classifier.

## References

[1] Billsus, D. & Pazzani M. (1998) Learning Collaborative Filters. In *Proceedings of the 15th International Conference on Machine Learning,* San Francisco, CA., Morgan Kaufmann Publishers.

[2] Breese, J., Heckerman, D., Kadie, C. (1998) Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, Madison, WI., Morgan Kaufmann Publisher.

[3] Domingos, P. & Pazzani M.(1997) On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. *Machine Learning*, 29, 103-130.

[4] Gupta, D., Digiovanni, M., Narita, H., Goldberg, K. (1999) Jester 2.0: A New Linear-Time Collaborative Filtering Algorithm Applied to Jokes. *Workshop on Recommender Systems Algorithms and Evaluation, 22nd International Conference on Research and Development in Information Retrieval*, Berkeley, CA.

[5] Herlocker, J., Konstan, J., Borchers, A., Riedl, J. (1999) An Algorithmic Framework for Performing Collaborative Filtering. In *proceedings of 22nd International Conference on Research and Development in Information Retrieval* 230-237, Berkley, CA., ACM Press.

[6] Hill, W., Stead, L., Rosenstein, M., Furnas, G.(1995) Recommending and Evaluating Choices in a Virtual Community of Use. In *Proceedings of the Conference on Human Factors in Computing Systems*, 194-201, Denver, CO., ACM Press.

[7] Lewis, D. & Gale, W. A. (1994) A sequential algorithm for training text classifiers. In *Proceedings of 17th International Conference on Research and Development in Information Retrieval*, 3-12, London, Springer-Verlag.

[8] Lewis, D. (1998) Naïve (Bayes) at forty: The independence assumption in information retrieval. In *Proceedings of the Tenth European Conference on Machine Learning.*

[9] McCallum, A. & Nigam, K. (1998) A Comparison of Event Models for Naïve Bayes Text Classification. *American Association for Artificial Intelligence (AAAI) Workshop on Learning for Text Categorization*.

[10] McJonese, P. (1997). EachMovie collaborative filtering data set. DEC Systems Research Center.

[11] Mitchell, T., (1997) *Machine Learning*. MacGraw-Hill, New York.

[12] Pazzani, M. & Billsus, D. (1997) Learning and Revising User Profiles: The identification of interesting web sites. *Machine Learning* 27, 313-331.

[13] Quinlan, J.R. (1986). Induction of decision trees. *Machine Learning* 1, 81-106.

[14] Resnick, P. & Varian, H. (1997) Recommender systems. *Communications of the ACM*, 40(3) 56-58.

[15] Resnick, P., Neophytos, I., Mitesh, S. Bergstrom, P. Riedl, J. (1994) GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of CSCW94: Conference on Computer Supported Cooperative Work*, 175-186, Chapel Hill, Addison-Wesley.

[16] Shardanand, U. & Maes, P. (1995) Social Information Filtering: Algorithms for Automating 'Word of Mouth'. In *Proceedings of the Conference on Human Factors in Computing Systems*, 210-217, Denver, CO., ACM Press.