Proc. 2nd International Workshop on Management of Information on the Web - Web Data and Text Mining (MIW'01)

Feature Weighting and Instance Selection for Collaborative Filtering*

Kai Yu[2], Zhong Wen[2], Xiaowei Xu[1], Martin Ester[2]
[1] *Information and Communications, Corporate Technology, Siemens AG*
[2] *Institute for Computer Science, University of Munich*
Xiaowei.Xu@mchp.siemens.de, {yu_k, wen, ester}@dbs.informatik.uni-muenchen.de

Abstract

Collaborative filtering uses a database about consumers' preferences to make personal product recommendations and is achieving widespread success in E-Commerce nowadays. In this paper, we present several feature-weighting methods to improve the accuracy of collaborative filtering algorithms. Furthermore, we propose to reduce the training data set by selecting only highly relevant instances. We evaluate various methods on the well-known EachMovie data set. Our experimental results show that mutual information achieves the largest accuracy gain among all feature-weighting methods. The most interesting fact is that our data reduction method even achieves an improvement of the accuracy of about 6% while speeding up the collaborative filtering algorithm by a factor of 15.

1. Introduction

The Internet is increasingly used as a channel for sales and marketing. More and more people purchase products through the Internet. One main problem that the customers face is how to find the product they like from millions of products. For the vendor, again, it is crucial to find out about the customers' preferences for products. Collaborative filtering or recommender systems have emerged in response to these problems[1] [6][10].

Collaborative filtering accumulates a database of consumers' product preferences, and then uses them to make customer-tailored recommendations for products such as clothing, music, books, furniture, and movies. The consumer's preference can be either explicit votes or implicit usage/purchase history. Collaborative filtering can help E-commerce in converting web surfers into buyers by personalization of the web interface. It can also improves cross-sell by suggesting other products the consumer might be interested in. In a world where an E-commerce site's competitors are only a click or two away, gaining customer loyalty is an essential business strategy. Collaborative filtering can improve the loyalty by creating a value-added relationship between supplier and consumer.

Collaborative filtering has been very successful in both research and practice. However, there still remain important research issues in overcoming two fundamental challenges for collaborative filtering [8].

The first challenge is to improve the scalability of the collaborative filtering algorithms. Existing collaborative filtering algorithms can deal with thousands of consumers within a reasonable time, but the demand of modern E-Commerce systems is to handle tens of millions of consumers.

The second challenge is to improve the quality of the recommendations for the consumers. Consumers need recommendations they can trust to help them find products they will like. If a consumer trusts a recomender system, purchases a product, but finds out he does not like the product, the consumer will be unlikely to use the recommender systems again.

In this paper, we present different feature weighting methods to improve the accuracy of collaborative filtering algorithm. Furthermore, we introduce a relevance measure of an instance to the target and propose to reduce the training data set by selecting only highly relevant instances.

In section 2, we briefly introduce collaborative filtering algorithms. We present different feature weighting methods including inverse user frequency, entropy and mutual information in section 3. We propose a mutual information based data reduction method for collaborative filtering in section 4. The empirical evaluation of these methods and results are reported in section 5. The paper ends with a summary and some interesting future work.

2. Collaborative Filtering

The task in collaborative filtering is to predict the preference of an active consumer to a given product based on a database of consumer' product preferences. There are two general classes of collaborative filtering algorithms: memory-based methods and model-based methods.

Memory-based algorithm [6][10] is the most popular prediction technique in collaborative filtering

* The work was performed in Cooperate Technology, Siemens AG. The contact author is Xiaowei Xu: Xiaowei.Xu@mchp.siemens.de

applications. The basic idea is to compute the active consumer's predicted vote of a product as a weighted average of the votes given to that product by other consumers. Specifically, the prediction $P_{a,j}$ of the active consumer *a* on product *j* is given by:

$$P_{a,j} = \bar{v}_a + k\sum_{i=1}^{n} w(a,i)(v_{i,j} - \bar{v}_i) \qquad (2.1)$$

where *n* is the number of the consumers who rated product *j*. $\bar{v}_i$ is the mean vote for consumer *i*. $v_{i,j}$ is the vote cast by *i* on *j*. $w(a,i)$is the similarity measure between *a* and *i*. *k* is a normalizing factor such that the absolute values of the weights sum to unity. There are two popular similarity measures: a person correlation coefficient and cosine vector similarity. Since the correlation-based algorithm outperforms the cosine vector based algorithm[1], we use the former one as the similarity measure. The person correlation coefficient [6] between consumer *a* and *i* is defined as:

$$w(a,i) = \frac{\sum_j \left(v_{a,j} - \bar{v}_a\right)\left(v_{i,j} - \bar{v}_i\right)}{\sqrt{\sum_j \left(v_{a,j} - \bar{v}_a\right)^2 \sum_j \left(v_{i,j} - \bar{v}_i\right)^2}} \qquad (2.2)$$

Memory-based methods have the advantages of being able to rapidly incorporate the most up-to-date information and relatively accurate prediction [1], but they suffer from poor scalability for large numbers of consumers. This is because the search for all similar consumers is slow in large databases.

Model-based collaborative filtering, in contrast, uses the consumers' preference database to learn a model, which is then used for predications. The model can be built off-line over several hours or days. The resulting model is very small, very fast, and essentially as accurate as memory-based methods [1]. Model-based methods may prove practical for environments in which consumer preferences change slowly with respect to the time needed to build the model. Model-based methods, however, are not suitable for environments in which consumer preference models must be updated rapidly or frequently.

In this paper, we will focus on memory-based algorithms and present some new methods to improve the scalability and the accuracy.

3. Feature Weighting Methods

As indicated before, collaborative filtering is built on the assumption that a good way to predict the preference of the active consumer for a target product is to find other consumers who have similar preferences, and then use those similar consumers' preferences for that product to make a prediction. The similarity measure is based on preference patterns of consumers. Therefore, a consumer's votes on the product set except the target product can be regarded as features of this consumer. Hence, introduction of some feature weighting methods may be useful to improve the accuracy of prediction. Through weighting, we can focus on the *good* products while removing *bad* ones or reducing their impacts. Votes on a 'good product' are highly relevant to the preference for the target product, while a 'bad product' is irrelevant or noisy in prediction for the target product. Such weighting methods can be derived from psychological and statistical observations. When using weight the similarity measures between consumers are modified as follows:

$$w(a,i) = \frac{\sum_j w_j^2 \left(v_{a,j} - \bar{v}_a\right)\left(v_{i,j} - \bar{v}_i\right)}{\sqrt{\sum_j w_j^2 \left(v_{a,j} - \bar{v}_a\right)^2 \sum_j w_j^2 \left(v_{i,j} - \bar{v}_i\right)^2}} \qquad (3.1)$$

where w_j represent the weight of product *j* with respect to the target product.

3.1 Inverse User Frequency

In applications of vector similarity in information retrieval, word frequencies are typically modified by the *inverse document frequency* [7]. The idea is to reduce weights for commonly occurring words, capturing the intuition that they are not useful in identifying the topic of a document, while words that occur less frequently are more indicative of the topic. Bresse et al [1] applied an analogous transformation to votes in a collaborative filtering database, which is termed inverse user frequency. The idea is that universally liked products are not as useful in capturing similarity as less common products. So inverse user frequency weight is defined as follows:

$$w_j = \log\frac{n}{n_j} \qquad (3.2)$$

where n_j is the number of consumers who have voted for product *j*, and *n* is the total number of consumers in the database. Note that if everyone has voted on product *j*, then the weight is zero. However, if in a database every product received about the same number of votes, this weighting method can not make sense.

3.2 Entropy

The concept of entropy was introduced as a measure of uncertainty of a random variable [9]. The diversity (or distribution) of consumer' votes to a specific product will be apparently meaningful in collaborative filtering. Let's consider a special case, if all the consumers give a very high vote on a product, then it is indicated that all the votes on this product will make no sense in computing similarity between consumers, because it can't tell any distinction among consumers. But if all the votes on a item are very diverse, almost identically distributed over

the range of the vote, then all the votes on this product will be very indicative in capturing the bias of consumers. Based on the above intuition, we propose an entropy-based weighting method in collaborative filtering:

$$w_j = H_j / H_{j,\max} \tag{3.3}$$

where $H_j = -\sum_i p_{i,j} \cdot \log_2 p_{i,j}$

In eq.(3.3) H_j is the entropy of product j, $p_{I,j}$ is the probability of votes on product j valued i, and $H_{j,max}$ represents the maximum entropy which assumes the distributions over all classes of vote are identical. This term is introduced to avoid the impact of different discrete vote ranges for different products. Thus, a large value of w_j means diverse preference for product j, and hence more emphases should be put on those votes for j in prediction. However, the proposed entropy-based weighting scheme might encounter the risk that there is no significant difference of the entropy from product to product. In the case of movie recommendation it is quite possible that the people's tastes towards every specific movie are all very diverse. In such a case, w_j is close to 1 for most of the movies and the entropy-based feature weighting can't result in any impressive improvement.

3.3 Mutual Information

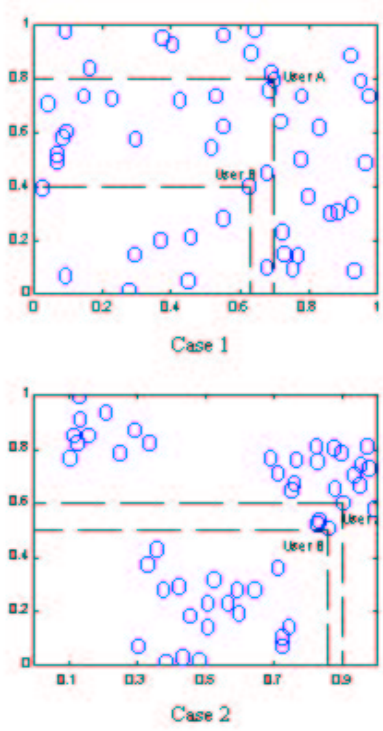
The two feature-weighting methods mentioned above are derived from the characteristics of single products. But our task is from some knowledge about other features to make a prediction for the target. So a better way should be to explore some kind of internal connection between features and the target. If the votes of the target product are found to be highly dependent on the votes of product j, we should assign a large weight to j.

Example 1. If 50 consumers/users give votes for two movies, i and j, and vote takes the value from 0 to 1, let us consider two different situations, case 1 and case 2 respectively, as shown in fig.1. In case 1, we find consumers are nearly uniformly distributed in the movie-movie vote space. If A and B are two arbitrary users who both have close interests to movie i, it does not necessarily indicate that they have also similar preferences for movie j. But in case 2, the situation is quite different. We can find for those consumers who dislike movie i, movie j always is their favorite. While those consumers who like movie i always rate the other one just above the average. In summary, in the second case movie i should play an important role in inferring some consumer's preference for movie j, while in case 1 it is not so useful.

Formally, the dependence of product j on i may be defined by a conditional probability:

$$p(|v_{j,u1} - v_{j,u2}| < e \,\big|\, |v_{i,u1} - v_{i,u2}| < e) \tag{3.4}$$

where $u1$ and $u2$ represent two arbitrary consumers and e is a threshold. If the difference between two votes is below e, those two votes are regarded to be 'close'. The above conditional probability indicates the probability of the case that two arbitrary consumers have close preference for product j given the condition that those two



consumers have close preference for product i.

Figure 1. Consumer in example 1

To apply dependence as a weighting scheme in collaborative filtering, we could calculate it according to formula (3.4). However, this would be very expensive since its runtime complexity is $O(n^2m^2)$ if n is the number of consumers and m the number of products. Instead, we will approximate dependence by the mutual information between a feature and the target. We will see below that this approximation behaves well and is significantly more efficient to calculate.

In information theory, mutual information represents a measure of statistic dependence between two random variables X and Y with associated probability distributions $p(x)$ and $p(y)$ respectively. Following Shannon [9], the mutual information between X and Y is defined as:

$$I(X;Y) = \sum_x \sum_y p(x,y) \log \left(\frac{p(x,y)}{p(x)\,p(y)} \right) \tag{3.5}$$

Furthermore, mutual information can be equivalently transformed into the following formulas:

$$I(X;Y) = H(X) - H(X \mid Y) \quad (3.6)$$

$$I(X;Y) = H(Y) - H(Y \mid X) \quad (3.7)$$

$$I(X;Y) = H(X) + H(Y) - H(X,Y) \quad (3.8)$$

where $H(X)$ is the entropy of X, $H(X|Y)$ is the conditional entropy of X given Y and $H(X,Y)$ is the joint entropy of two random variables. The definition of the conditional entropy, the joint entropy and the proof of the above equations can be found in [2]. The equations above indicate that mutual information also represents the reduction of entropy (uncertainty) of one variable given information of the other variable.

Theorem: Given two products i and j, as well as distributions of votes on them, $P(V_i)$ and $P(V_j)$. And e is the interval of discrete value for vote. If u1 and u2 are two arbitrary consumers who have voted for both products, then

$$\frac{d\left[p(|v_{j,u1} - v_{j,u2}| < e \;\middle|\; |v_{i,u1} - v_{i,u2}| < e)\right]}{d\left[I(V_j;V_i)\right]} > 0 \quad (3.9)$$

Proof:

Since $P(V_i)$ and $P(V_j)$ are given, we have:

$$d\left[I(V_j;V_i)\right] = d\left[H(V_j) - H(V_j|V_i)\right] = -d\left[H(V_j|V_i)\right] \quad (3.10)$$

Inequation (3.8) can be written as:

$$\frac{d\left[p(|v_{j,u1} - v_{j,u2}| < e \;\middle|\; |v_{i,u1} - v_{i,u2}| < e)\right]}{d\left[H(V_j|V_i)\right]} < 0 \quad (3.11)$$

Next, we have

$$H(V_j \mid V_i) = \sum_{v \in \aleph} p(V_i \equiv v) H(V_j \mid V_i \equiv v) \quad (3.12)$$

and

$$p(|v_{j,u1} - v_{j,u2}| < e \;\middle|\; |v_{i,u1} - v_{i,u2}| < e) = \frac{\sum_{v \in \aleph} p(V_i \equiv v)^2 p\left(|v_{j,u1} - v_{j,u2}| < e \mid v_{i,u1} = v_{i,u2} = v\right)}{\sum_{v \in \aleph} p(V_i \equiv v)^2} \quad (3.13)$$

where $\aleph$ is the set of all discrete votes. From eq. (3.12) and eq. (3.13) we can easily derive ineq. (3.11). Therefore, ineq.(3.9) holds. □

The above theorem clearly shows that large mutual information between the feature and the target means a high dependency between them. Therefore, it encourages us to propose mutual information as a weighting method in collaborative filtering.

$$w_j = I(V_j;V_t) \quad (3.14)$$

where V_j and V_t are the votes on product j and target product t respectively. According to eq. (3.8), we use the following equation to estimate the mutual information between two products:

$$I(V_j;V_t) = H(V_j) + H(V_t) - H(V_j,V_t) \quad (3.15)$$

where $H(V_j,V_t)$ is the joint entropy between two products. Since not all the consumers have voted for the two products, calculation is done over the overlap. If the average number of overlapping consumers between two products is n, and there are totally m products in the training data set, the computational complexity for calculating the mutual information between all pairs of products is $O(nm^2)$.

4. Selecting Relevant Instances

An interesting question is, since the number of recorded consumers is explosively increasing, how to speed up the prediction? To respond to this challenge, we propose a method to reduce the training data set by selecting only highly relevant instances. In our application the instance is the consumers in the preference database.

In collaborative filtering algorithm, the computational complexity is linear with respect to the number of consumers who cast a vote to the predicted product (n in eq. 2.1). Therefore, one way to speed up the process of recommendation is to reduce the number of consumers for every target product in the training data set. This can be done through random sampling or data focussing techniques[3][4]. However these methods have the problem that the quality of the prediction is reduced due to the loss of information.

We propose a data reduction method that can even improve the quality of the prediction. Intuitively, this data reduction works as follows: First, we pick up the consumers who have given votes to many products because of their low sparsity and clear profile. Secondly, we wish to select consumers whose votes are mainly over dependant products, since those products can provide more accurate information to infer a consumer's preference. Based on the above analysis, we use the following measure to rank the *relevance* of consumer i to target product t:

$$R_{i,t} = \log n_i \cdot \frac{\sum_{j \in M_i, j \neq t} I(V_j;V_t)}{n_i - 1} \quad (3.15)$$

where n_i is the number of the votes cast by i. M_i is the set of products voted by i. For every product in the training data set, we rank consumers who cast a vote to the product according the relevance (eq. 3.15) and only the top k% of the ranking list will be used in the prediction. The rest (1-k)% will be removed from the training data set. In this case, the *selection rate* is k% and the *reduction rate* is (1-k)%.

5. Experimental Evaluation

In this section, we report results of an experimental evaluation of our proposed techniques. We describe the data set used, the experimental methodology, as well as

the performance improvement compared with traditional techniques.

5.1 The EachMovie Database

We ran experiments using data from the *EachMovie* collaborative filtering service, which was part of a research project at the Systems Research Center of Digital Equipment Corporation. The database contains votes from 72,916 users on 1,628 movies. User votes were recorded on a numeric six-point scale (We transfer it to 0, 1, 2, 3, 4, and 5).

Although data from 72,916 users is available, we restrict our analysis to 35,527 users who gave at least 20 votes over the totally 1623 movies. For those users whose vote number is less than 20, since their profiles are unclear, it is hard to be used in evaluation. Moreover, to speed up our experiments, we randomly selected 10,000 users from the 35,527 users and divided them into a training set (8000 users) and a test set (2000 users).

5.2 Metrics and Methodology

Since we are interested in a system that can accurately predict a consumer's vote on a specific product, we use the mean absolute error (MAE), where the error is the value of the differences between the actual vote and the predicted vote, to evaluate the quality of prediction. This metric has been widely used in previous work[1], [5], [6] and [10].

As in [1], we also employ two protocols, *All but One*, and *Given K*. In the first class, we randomly hide an existing vote for each test consumer, and try to predict its value given all the other votes the consumer has voted on. The All but One experiments investigate the algorithms' performance when given as much data as possible from each test consumer, and are indicative of what might be expected of the algorithms under steady state usage where the database has accumulated a fair amount of data about a particular consumer. The second protocol, *Given K*, randomly select *K* votes from each test consumer as the observed votes, and then attempts to predict the remaining votes. It looks at consumers with less data available, and examines the performance of the algorithms when there is relatively little known about an active consumer. Its results show the performance of algorithms during the startup period, when a consumer is new to a particular collaborative filtering recommender.

5.3 Results

As shown in Fig.2, we investigate the accuracy of collaborative filtering using different feature weighting methods. The experiments were conducted for training set with 200, 500, 1000, 2000, 5000 and finally 8000 consumers. Our result show that mutual information based weighting achieves the best accuracy, yielding an improvement of about 5% compared to the standard method without feature weighting. Entropy based method, on the other hand obtain only a slight improvement. This can be explained by the fact that the variance of entropy across movies is not very large. We also find that weighting by the inverse user frequency even reduces the accuracy of prediction.

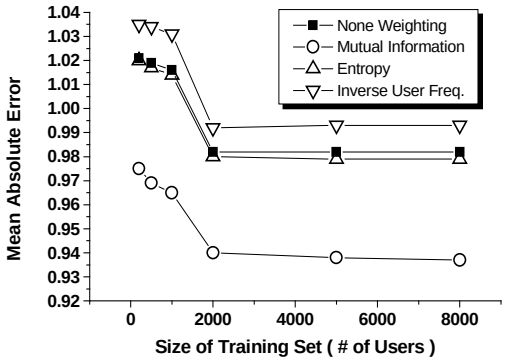


Figure 2. *All but One* results of feature weighting in different training sizes

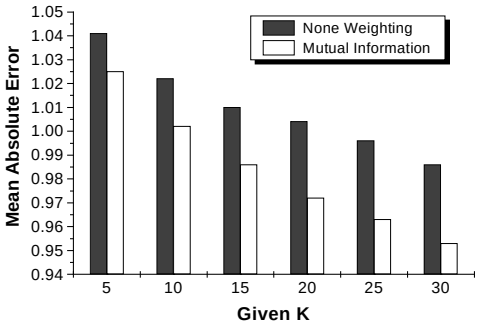


Figure 3. *Given K* results of mutual information weighting method

Fig.3 shows results under the protocols of *Given 5, 10, 15, 20, 25, 30*. In the six cases, mutual information weighting results in an improved accuracy. The improvement of MAE varies from 1.5% to 4.5%. The results indicate the more we know about the active consumer, the more improvement can be achieved by our weighting scheme.

We also evaluated the performance of our method of selecting relevant instances. The outcomes are given in Fig. 4 and Fig. 5. As described in section 4, we sort consumers in descending order of their relevance to each movie, and select highly relevant consumers for the prediction in different selection rates of 3.13%, 6.25%, 12.5%, 25%, 62.5% and 100%. The results are compared with the outcomes of random sampling. The proposed method outperforms random sampling in accuracy, and

the combination with mutual information based feature weighting results in further 4~5% improvement of mean absolute error. As shown in Figure 5, the computational complexity is linear with respect to the number of consumers in the training data set. For example, if we select 6.25% of the training size, the average prediction time for each vote is reduced from 399 ms to 27.5 ms.

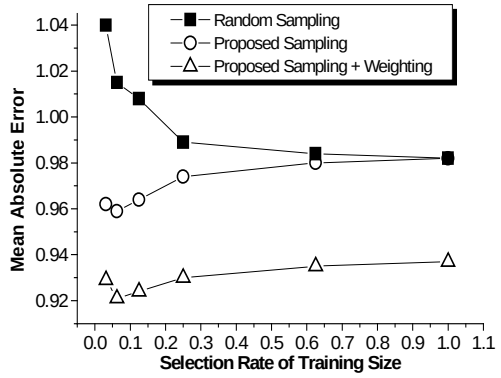


Figure 4. *All But One* performance for different selection rates

Moreover, Figure 4 shows that there is an optimal selection rate with respect to the accuracy, which is 6.25%. To conclude, we can achieve over 6% improvement in accuracy by using only 6.25% of the whole training data set while at the same time reducing the runtime by a factor of 15. We think it is due to the existence of irrelevant consumers in the whole training set and those irrelevant consumers are the noise for the target product.

6. Conclusion

In this paper, we present different feature weighting methods to improve the accuracy of the memory-based collaborative filtering algorithm. Furthermore, we introduce a relevance measure of an instance to the target and propose to reduce the size of training data set by selecting only highly relevant instances. We give an empirical evaluation of different feature weighting methods. Our results show that mutual information achieves the best accuracy. Our data reduction method can significantly reduce the size of the training data set and speed up the collaborative filtering algorithm. The most interesting fact is that our method even achieves an improvement on the accuracy of about 6% at a reduction rate of 94%, while random sampling decreases the accuracy by 4%.

Our result shows that feature weighting and selecting relevant instances are very promising methods for data mining. In the future, we will apply our methods to model-based collaborative filtering algorithms. We will also investigate the performance of our methods to other applications such as web page usage mining and text mining.

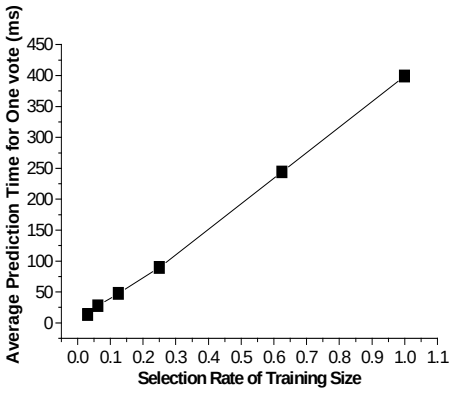


Figure 5. Prediction time for different selection rates

7. References

[1] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical Analysis of Predictive Algorithms for Collaborative Filtering", In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, 1998.

[2] G. Deco, and D. Obradovic, *An Information-Theoretic Approach to Neural Computing*, Spinger-Verlag Inc., New York, 1996.

[3] M. Ester, H.-P. Kriegel, and X. Xu, "Knowledge Discovery in Large Spatial Databases: Focusing Techniques for Efficient Class Identification", In *Proc. of 4th Int. Symp. on Large Spatial Databases*, Portland, ME, 1995, also in *Lecture Notes in Computer Science, Vol. 951*, Springer, 1995, pp.67-82.

[4] M. Ester, H.-P. Kriegel, and X. Xu, "A Database Interface for Clustering in Large Spatial Databases", In *Proc. 1st Int. Conf. on Knowledge Discovery and Data Mining (KDD95)*, Montreal, Canada, 1995, pp. 94-99.

[5] W. Hill, L. Stead, M. Rosenstein, and G. Furnas, "Recommending and evaluating choices in a Virtual Community of Use", In *Proceedings of CHI'95*.

[6] P. Resnick, N. Iacovou, M. Sushak, P. Bergstrom, and J. Riedl, "GroupLens: An Open Architecture for Collaborative Filtering of Netnews", In *Proceedings of the 1994 Computer Supported Collaborative Work Conference*.

[7] G. Saloton, and M. McGill, *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, 1983.

[8] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl, "Analysis of Recommender Algorithms for E-Commerce", In *Proceedings of ACM E-Commerce 2000 Conference*.

[9] C. E. Shannon, "A Mathematical Theory of Communication", *Bell Sys. Tech. Journal*, vol. 27, 1948

[10] U. Shardanand, and P. Maes, "Social Information filtering Algorithms for Automating 'Word of Mouth'", In *Proceedings of CHI'95*.