A Scalable, Accurate Hybrid Recommender System

Mustansar Ali Ghazanfar and Adam Prugel-Bennett
School of Electronics and Computer Science
University of Southampton
Highfield Campus, SO17 1BJ, United Kingdom
Email: {mag208r, adb}@ecs.soton.ac.uk

***Abstract*—Recommender systems apply machine learning techniques for filtering unseen information and can predict whether a user would like a given resource. There are three main types of recommender systems: collaborative filtering, content-based filtering, and demographic recommender systems. Collaborative filtering recommender systems recommend items by taking into account the taste (in terms of preferences of items) of users, under the assumption that users will be interested in items that users similar to them have rated highly. Content-based filtering recommender systems recommend items based on the textual information of an item, under the assumption that users will like similar items to the ones they liked before. Demographic recommender systems categorize users or items based on their personal attribute and make recommendation based on demographic categorizations. These systems suffer from scalability, data sparsity, and cold-start problems resulting in poor quality recommendations and reduced coverage. In this paper, we propose a unique cascading hybrid recommendation approach by combining the rating, feature, and demographic information about items. We empirically show that our approach outperforms the state of the art recommender system algorithms, and eliminates recorded problems with recommender systems.**

***Keywords*-Recommender systems; collaborative filtering; content-based filtering; demographic recommender system.**

I. Introduction

There has been an exponential increase in the volume of available digital information, electronic resources, and on-line services in recent years. This information overload has created a potential problem, which is how to filter and efficiently deliver relevant information to a user. This problem highlights a need for information extraction systems that can filter unseen information and can predict whether a user would like a given resource. Such systems are called *recommender systems*.

Let $M = \{\ m_1, m_2, \cdots, m_x\ \}$ be the set of all users, $N = \{\ n_1, n_2, \cdots, n_y\ \}$ be the set of all possible items that can be recommended, and r_{m_i,n_j} be the rating of user m_i on item n_j. Let u be a utility function that measures the utility of item n_j to user m_i, i.e.

$$u : M \times N \rightarrow R, \tag{1}$$

where R is a totally ordered set. Now for each user $m_i \in M$, the aim of a recommender system is to choose that item $n'_j \in N$ which maximizes the user's utility [1]. We can specify this as follows:

$$n'_{j_{m_i}} = \arg\max_{n_j \in N} u(m_i, n_j)\ : \forall_{m_i \in M}, \tag{2}$$

where the utility of an item is application dependent.

Collaborative Filtering (CF) systems can be classified into two sub-categories: memory-based CF and model-based CF [1]. Memory-based approaches make a prediction by taking into account the entire collection of previous rated items by a user; examples include user-based CF algorithms [2]. Model-based approaches learn a model from collection of ratings and use this model for making prediction; examples include item-based CF [3] and Singular Value Decomposition (SVD) based models [4]. Model-based approaches are more scalable than user-based approaches.

There are two potential problems with the recommender systems. One is the *scalability*, which is how quickly a recommender system can generate recommendation, and the second is to ameliorate the *quality* of the recommendation for a customer. Pure CF recommender systems produces high quality recommendation than those of pure content-based and demographic recommender systems, however, due to the *sparsity*[1], they can not find similar items or users using rating correlation, resulting in poor quality predictions and reduced *coverage*[2]. Furthermore, All individual systems fail in cold-start [1] problems.

In this paper, we propose a hybrid scheme (namely $Boosted_{RDF}$) which produces accurate and practical recommendation and can be used in cold-start scenarios. Our proposed scheme is based on a cascading hybrid recommendation technique[3] [5] that build item models based on item's rating, feature, and demographic information; and generates more accurate prediction than available state of the art recommender algorithms. We evaluate our algorithm on MovieLens[4] and FilmTrust[5] datasets.

II. Background

A. Item-Based Collaborative Filtering Recommender Systems: Item Rating Information

Item-based CF recommender systems [3] build a model of item similarities using an off-line stage. There are three main steps in these systems as follows:

[1]The percentage of ratings assigned by users is very small as compared to the percentage of ratings the system has to predict.

[2]The percentage of the items that can be recommended from all available items in the system.

[3]In cascading hybrid recommender systems, a recommendation technique is applied to produce a coarse candidate list of items for recommendation that are refined by applying other recommendation techniques.

[4]www.grouplens.org/node/73.

[5]http://trust.mindswap.org/FilmTrust.

Table I
A SAMPLE OF FUNCTIONS (F) WITH $k = 20$

Function #	Function (f)	MAE (ML)	MAE) (FT)
1	$R_{I\,Sim}$ (Item-based CF)	0.791	1.442
2	$F_{I\,Sim}$	0.787	1.436
$\vdots$	$\vdots$	$\vdots$	$\vdots$
32	$F_{I\,Sim} + R_{D\,Sim} + D_{D\,Sim}$	**0.736**	**1.379**
$\vdots$	$\vdots$	$\vdots$	$\vdots$
83	$R_{I\,Sim} + F_{I\,Sim} + D_{I\,Sim}$ $R_{D\,Sim} + F_{D\,Sim} + D_{D\,Sim}$	0.834	1.451

1- All items rated by an *active user*[6] are retrieved.
2- *Target item*[7]'s similarity is computed with the set of retrieved items. A set of k most similar items, also known as *neighbours* of the target item with their similarities are selected.
3- *Prediction* for the target item is made by computing the weighted sum of the active user's rating on the k most similar items.

B. Content-Based Recommender Systems: Item Feature Information

Content-based recommender systems [1] recommend items based on the textual information of an item. In these systems, an item of interest is defined by its associated features, for instance, NewsWeeder [8], a newsgroup filtering system uses the words of text as features.

We downloaded information about movies from IMDB[8], and applied $TF-IDF$ [8] approach for extracting the features from the information about each movie. After stop word removal and stemming, we constructed a vector of keywords, tags, directors, actors/actresses, and user reviews given to a movie in IMDB. Furthermore, we leverage WordNet using Java Wordnet Interface[9] for overcoming the synonym problems between features while finding the similarities among (text) features.

C. Demographic Recommender Systems: Item Demographic Information

Demographic recommender systems [6], [5] categorize users or items based on their personal attributes and make recommendation based on demographic categorizations. In our work, we used genre information about a movie as its demographic information and constructed a vector as used in [7].

III. EXPERIMENTAL EVALUATION

A. Dataset

We used MovieLens (ML) and FilmTrust (FT) datasets for evaluating our algorithm. MovieLens dataset contains 943 users, 1682 movies, and 100 000 ratings on an integer scale of 1 (bad) to 5 (excellent). MovieLens dataset has been used in many research projects [3], [4], [7]. The sparsity of this dataset is 93.7% $\left(1-\frac{non\ zero\ entries}{all\ possible\ entries} = 1-\frac{100000}{943\times1682} = 0.937\right)$.

we created the second dataset by crawling the FilmTrust website. The dataset retrieved (on 10th of March 2009) contains 1592 users, 1930 movies, and 28 645 ratings on a floating point scale of 1 (bad) to 10 (excellent). The sparsity of this dataset is 99.06%[10].

B. Metrics

Our specific task in this paper is to predict scores for items that already have been rated by actual users, and to check how well this prediction helps users in selecting high quality items. Keeping this into account, we use *Mean Absolute Error (MAE)* and *ROC (Receiver Operating Characteristic) sensitivity*.

MAE measures the average absolute deviation between a recommender system's predicted rating and a true rating assigned by the user. It is computed as follows:

$$MAE = \frac{\sum_{i=1}^{N} |r_{p_i} - r_{a_i}|}{N},$$

where r_{p_i} and r_{a_i} are the predicted and actual values of a rating respectively, and N is the total number of items that have been rated. It has been used in [2], [3].

ROC sensitivity measures the probability with which a system accept a good item. The ROC sensitivity ranges from 1 (perfect) to 0 (imperfect). For MovieLens dataset, we consider an item good if the user rated it with a score of 4 or higher and bad otherwise. Similarly, for FilmTrust dataset, we consider an item good if the user rated it with a score of 7 or higher and bad otherwise. It has been used in [9].

Furthermore, we used *coverage* that measures how many items a recommender system can make recommendation for. It has been used in [10].

IV. PROPOSED ALGORITHM

Let m_a, n_t, R, D, F be the active user, target item, user-item rating matrix [1], demographic vector for an item n_j, and feature vector for an item n_j respectively. Let $R_{I\,Sim}$, $D_{I\,Sim}$, and $F_{I\,Sim}$ represent the rating, demographic, and feature similarity among the items. Furthermore, let $R_{D\,Sim}$, $D_{D\,Sim}$, and $F_{D\,Sim}$ represent the rating similarity among candidate items found after applying the feature correlation among all items, demographic similarity among candidate items found after applying the feature correlation among all items, and feature similarity among candidate items found after applying the rating correlation among all items.

The proposed algorithm can be summarized as follows:

Step-1: Compute the similarity between items using rating data, demographic data, and feature data, and store

[6]The user for whom recommendation are generated.
[7]The item a system wants to recommend.
[8]www.imdb.com.
[9]http://projects.csail.mit.edu/jwi.

[10]Both datasets can be downloaded from: https://sourceforge.net/projects/hybridrecommend.

Table II
A SAMPLE OF PARAMETER SET WITH $k = 20$

Parameter Set #	α	β	γ	**MAE (ML)**	**MAE (FT)**
1	0.1	0.1	0.8	0.739	1.381
⋮	⋮	⋮	⋮	⋮	⋮
29	**0.5**	**0.3**	**0.2**	**0.732**	1.378
⋮	⋮	⋮	⋮	⋮	⋮
35	**0.7**	**0.2**	**0.1**	0.738	**1.373**
36	0.8	0.1	0.1	0.741	1.379

this information. Adjusted cosine similarity [3] between two items is used for measuring the similarity over rating data. Vector similarity [2] between two items is used for measuring the similarity using demographic and feature vectors.

Step-2: Boosted similarity $Boosted_{Sim}$ is defined by a function, f_{max} that combines R_{ISim}, D_{ISim}, F_{ISim}, R_{DSim}, D_{DSim}, and F_{DSim} over set of items in the training set. This function uses (5) for making prediction, and tries to maximize the utility. Formally, it can be specified as follows:

$$f_{max} = \underset{f \in F}{\operatorname{argmax}}\, u(m_i, n_j) \ : \forall m_i \in M^T, \forall n_j \in N^T. \tag{3}$$

Equation (3) tells us to choose that function which maximizes the utility (i.e. reduces the *MAE*) of all users (M^T) over set of items (N^T) in the training set. Table I gives the different combination of functions checked over the training set with their respective lowest *MAE* observed. It shows that a cascading hybrid setting in which rating and demographic correlation are applied over the candidate neighbour items found after applying the feature correlation gives the minimum error.

Let $C_{n_t} = \{\ c_1, c_2, \cdots, c_k\ \}$ be the set of k candidate neighbours found after applying the feature similarity. We define the boosted similarity $Boosted_{Sim}$ by a linear combination of F_{ISim}, R_{DSim}, and D_{DSim} over the set of items in the training set as follows:

$$Boosted_{Sim}(n_t, c_i) = \alpha \times F_{ISim} + \beta \times R_{DSim} + \gamma \times D_{DSim}, \tag{4}$$

where, α, β, and γ parameters represent the relative impact of three similarities. We assume $\alpha + \beta + \gamma = 1$ without the loss of generality.

Step-3: Predicting the rating P_{m_a,n_t} for an active user m_a on target item n_t is made by using the following formula:

$$P_{m_a,n_t} = \frac{\sum_{i=1}^{k}\Big(Boosted_{Sim}(n_t, c_i) \times r_{m_a,c_i}\Big)}{\sum_{i=1}^{k}\Big(|Boosted_{Sim}(n_t, c_i)|\Big)}. \tag{5}$$

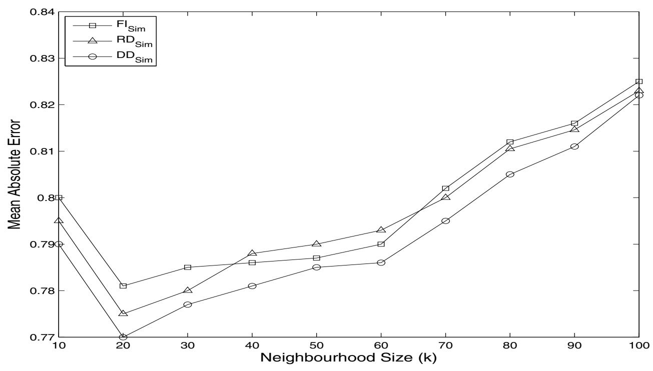

Figure 1. Determining the optimal value of neighbourhood size, k (MovieLens).

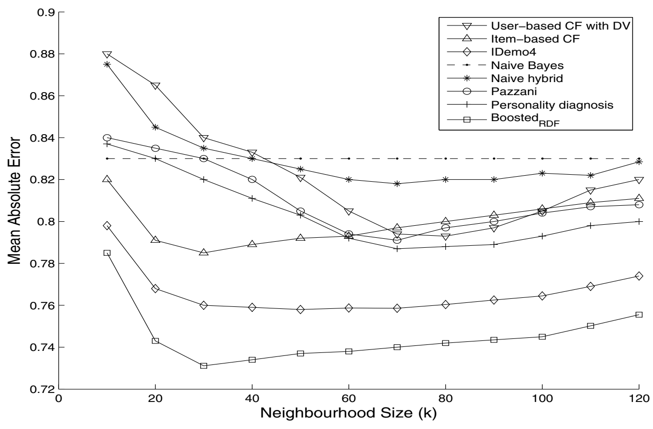

Figure 2. MAE of the proposed algorithm with others, against various neighbourhood sizes (MovieLens).

Equation (5) is the same as used by [3] except the rating similarity function has been replaced by $Boosted_{Sim}$.

V. Result and Discussion

We randomly selected 20% ratings of each user as the test set and used the remaining 80% as training set. We further subdivided our training set into a test set and training set for measuring the parameters sensitivity. For learning the parameters, we conducted 5-fold cross validation on the 80% training set, by randomly selecting the different test and training set each time, and taking the average of results.

We compared our algorithm with seven different algorithms: user-based CF using Pearson correlation with default voting (DV) [2], item-based CF using adjusted-cosine similarity [3], a hybrid recommendation algorithm, IDemo4, proposed in [7], a Naive Bayes classification approach using item features information, a naive hybrid approach for generating recommendation[11], the personality diagnosis algorithm [11] for making probabilistic recommendations, and a hybrid recommendation algorithm used by Pazanni [6]. Furthermore, we tuned all algorithms for the best mentioning parameters.

[11]We take average of the prediction generated by a pure content-based and a pure user-based CF.

Table III
A COMPARISON OF THE PROPOSED ALGORITHM WITH EXISTING IN TERMS OF COST (BASED ON [4], [11]), ACCURACY METRICS, AND COVERAGE

Algorithm	**On-line Cost**	**Best *MAE* (ML)**	**Best *MAE* (FT)**	***ROC Sensitivity* (ML)**	***ROC Sensitivity* (FT)**	***Coverage* (ML)**	***Coverage* (FT)**
User-based with DV	NM^2	0.791	1.441	0.401	0.643	99.424	93.611
Item-based	N^2	0.789	1.439	0.383	0.621	99.221	92.312
IDemo4	N^2	0.768	1.415	0.430	0.644	99.572	94.441
$Boosted_{RDF}$	N^2	0.725	1.362	0.562	0.756	100	99.195
Naive Bayes	$M(NP)$	0.831	1.470	0.622	0.835	100	99.997
Naive hybrid	$NM^2 + M(NP)$	0.822	1.462	0.526	0.726	100	99.991
Pazzani	NPM^2	0.793	1.440	0.552	0.739	99.920	99.031
Personality diagnosis	NM	0.785	1.432	0.521	0.732	99.142	94.232

A. Locating the Optimal Value of Neighbourhood Size (k)

We varied the number of neighbours for an active user, from 0 to 100 and computed the corresponding *MAE* for F_{ISim}, R_{DSim}, and D_{DSim}. The results are shown in Fig. 1.

Fig. 1 shows that for MovieLens dataset *MAE* is minimum for $k = 20$[12]. We choose the neighbourhood size to be 20 for further experiments.

B. Learning the Optimal Values of Parameters (α, β, γ)

The 36 parameter sets were generated by producing all possible combination of parameters values, ranging from 0.1 to 1.0 with differences of 0.1. Table II presents a sample of the parameter sets learned. The parameters sets $\alpha = 0.5, \beta = 0.3, \gamma = 0.2$, and $\alpha = 0.7, \beta = 0.2, \gamma = 0.1$ gave the lowest *MAE* in the case of MovieLens and FilmtTrust dataset respectively. It is worth noting that combined similarity depends heavily on feature similarity, i.e. α. Furthermore, the values of parameters are found different for MovieLens and FilmTrust dataset, which is due to the fact that both dataset have different density, rating distribution, and rating scale.

C. Comparison of the Proposed Algorithm with Others

1) Performance Evaluation in Terms of MAE: Fig. 2 shows that our algorithm significantly outperforms other algorithms[13]. Similar results were observed in the case of FilmTrust dataset. We can conclude from the results that clusters of items found after applying $F_{ISim}, R_{DSim}, D_{DSim}$ have complementary role for prediction generation.

2) Comparison of MAE, ROC Sensitivity, Coverage, and On-line Cost of the Proposed Algorithm with Others: Table III shows the on-line cost (in the worst case) of each algorithm, with the corresponding lowest *MAE*, *ROC sensitivity*, and *coverage*. Here, P is the number of features against a training example (i.e. features against a movie). It is worth noting that for FilmTrust dataset, ROC sensitivity is higher, for all algorithms in general, as compared to the MovieLens dataset. We believe that it is due to the rating distribution[14]. Furthermore, the coverage of the algorithms is much lower in the case of FilmTrust dataset, which is due to the reason that it is very sparse (99%). The table depicts that $Boosted_{DemoFeature}$ is scalable and practical as its on-line cost is less or equal to the cost of other algorithms.

Table IV
PERFORMANCE EVALUATION UNDER NEW ITEM COLD-START PROBLEM

Algorithm	**MAE1**		**MAE2**		**MAE5**	
	(ML)	**(FT)**	**(ML)**	**(FT)**	**(ML)**	**(FT)**
User-based CF with DV	1.64	2.67	1.23	2.22	0.95	1.94
Item-based CF	1.34	2.51	1.18	2.13	0.91	1.58
IDemo4	1.25	2.34	1.15	2.09	0.89	1.53
$Boosted_{RDF}$	0.98	1.62	0.85	1.57	0.82	1.42

3) Performance Evaluation Under New Item and New User Cold-Start Problems: When a new item is added to the system, then it is not possible to get rating for that item from significant number of users, and consequently the CF recommender system would not be able to recommend that item. This problem is called new item *cold-start problem* [1]. For testing our algorithm in this scenario, we selected 1000 random samples of user/item pairs from the test set. While making prediction for a target item, the number of users in the training set who have rated the target item were kept 1, 2, and 5. The corresponding *MAE*; represented by *MAE*1, *MAE*2, *MAE*5, is shown in the table IV. Table IV shows that proposed scheme works well in new item cold-start problem scenario, as it does not solely depend on the number of users who have rated the target item for finding the similarity.

For new user *cold-start problem* [2], where the profile of a user is incomplete, we use a linear regression model for finding an approximation of the active user's rating for an item. We use this rating instead of the active user's actual rating in (5) for prediction generation.

$$r'_{m_a,n_i} = \begin{cases} r_{m_a,n_i} : & \text{if active user rated more} \\ & \text{than } J \text{ movies} \\ r_{reg} : & \text{otherwise.} \end{cases} \qquad (6)$$

In 6, the choice of J came from the training set, which found to be 10 for MovieLens and 5 for FilmTrust dataset. Rating r_{reg} is found by a linear regression model: $R_s = \theta_1 R_t + \theta_2$, where R_s, and R_t are the vector of similar item and vector of target item respectively[15]. Parameters θ_1 and

[12]The size found to be between $20 - 30$ for FilmTrust dataset.

[13]In Fig. 2, the x-axis represents the number of neighbouring items in the case of item-based CF, IDemo4, and $Boosted_{DemoFeature}$; and the number of neighbouring users otherwise.

[14]In FilmTrust, majority of the users have rated the popular set of movies and their rating tends to match the average rating of the movies.

[15]These vector are made up of all user who have rated that item in the training set. For more information, refer to [3].

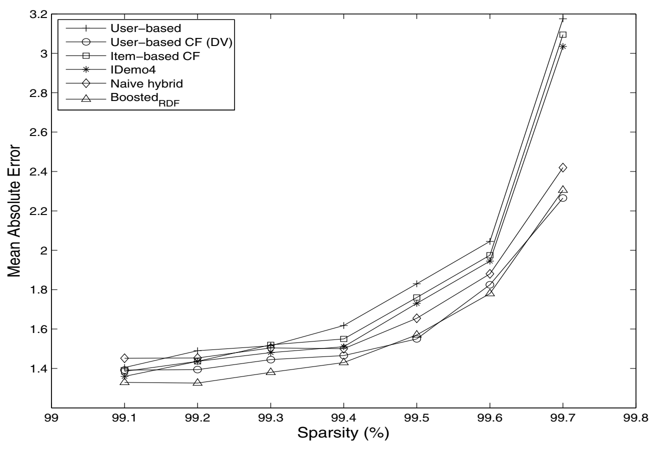

Figure 3. Performance of algorithms under different sparsity levels (FilmTrust).

θ_2 can found by both rating vectors. This model was used in [3] for overcoming the misleading similarities between items in the item-based CF. We show this model makes sense to apply only when we have incomplete user profile. The performance of the test set in the case of *Given 2* and *Given 5*[16] [2] protocol is given in the table V.

Table V shows that our algorithm with regression gave more accurate results than others. User-based CF with DV gave good results, as it assumes some default votes for items a user has not voted on.

4) Performance Evaluation Under Different Sparsity Levels: To check the effect of sparsity, we increased the sparsity level of the training set by dropping some randomly selected entries. Whereas, we kept the test set same for each sparse training set. We checked the performance of the proposed algorithm with those of pure user-based CF, user-based CF with DV, item-based CF, IDemo4, and a naive hybrid recommendation algorithm. Fig. 3[17] shows that performance does not degrades rapidly in the case of proposed algorithm. It is because; features of an item can still be used for finding similar items. Furthermore, synonym detection algorithm enrich item profiles while finding the similarity between the items.

VI. Conclusion

In this paper, we have proposed a cascading hybrid recommendation approach by combining the feature correlation with rating and demographic information of an item. We showed empirically that our approach outperformed the state of the art algorithms.

Acknowledgment

The work reported in this paper has formed part of the Instant Knowledge Research Programme of Mobile VCE, (the Virtual Centre of Excellence in Mobile & Personal Communications), www.mobilevce.com. The programme is co-funded by the UK Technology Strategy Boards Collaborative Research and Development programme. Detailed technical reports on this research are available to all Industrial Members of Mobile VCE. Thanks Martin for Providing the dataset.

[16]Given N means, active user has rated N items.

[17]Results were the same for MovieLens dataset. We show results for FilmTrust to analyse the performance of the algorithms under extreme sparsity (i.e. sparsity $> 99\%$).

Table V
Performance evaluation under new user cold-start problem

Algorithm	**MAE: Given 2**		**MAE: Given 5**	
	(ML)	**(FT)**	**(ML)**	**(FT)**
User-based CF with DV	1.06	2.03	0.86	1.48
Item-based CF	1.09	2.06	0.88	1.51
IDemo4	1.07	2.05	0.85	1.48
$Boosted_{RDF}$	1.06	2.01	0.83	1.47
$Boosted_{RDF_{Reg}}$	1.01	1.93	0.79	1.41

References

[1] A. T. Gediminas Adomavicius, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," IEEE Transactions on Knowledge and Data Engineering, vol. 17, 2005, pp. 734–749.

[2] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," Morgan Kaufmann, 1998, pp. 43–52.

[3] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-based collaborative filtering recommendation algorithms," in Proceedings of the 10th international conference on World Wide Web. ACM New York, NY, USA, 2001, pp. 285–295.

[4] M. Vozalis and K. Margaritis, "Using SVD and demographic data for the enhancement of generalized collaborative filtering," Information Sciences, vol. 177, no. 15, 2007, pp. 3017–3037.

[5] R. Burke, "Hybrid recommender systems: Survey and experiments, User Modeling and User-Adapted Interaction," vol. 12, no. 4, 2002, pp. 331–370.

[6] M. J. Pazzani, "A framework for collaborative, content-based and demographic filtering," Artificial Intelligence Review, vol. 13, no. 5–6, 1999, pp. 393–408.

[7] M. Vozalis and K. Margaritis, "On the enhancement of collaborative filtering by demographic data," Web Intelligence and Agent Systems, vol. 4, no. 2, 2006, pp. 117–138.

[8] K. Lang, Newsweeder: "Learning to filter netnews," in Proceedings of the Twelfth International Conference on Machine Learning, 1995.

[9] P. Melville, R. J. Mooney, and R. Nagarajan, "Content-boosted collaborative filtering for improved recommendations," in Eighteenth National Conference on Artificial Intelligence, 2002, pp. 187–192.

[10] L. G. T. Jonathan L. Herlocker, Joseph A. Konstan and J. T. Riedl, "Evaluating collaborative filtering recommender systems," ACM Transactions on Information Systems (TOIS) archive, vol. 22, 2004, pp. 734–749.

[11] D. Pennock, E. Horvitz, S. Lawrence, and C. Giles, "Collaborative filtering by personality diagnosis: A hybrid memory and model-based approach," in Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence, 2000, pp. 473–480.