Fulfilling the Needs of *Gray-Sheep* Users in Recommender Systems, A Clustering Solution

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

***Abstract*—Recommender systems apply data mining techniques for filtering unseen information and can predict whether a user would like a given item. This paper focuses on *gray-sheep* users problem responsible for the increased error rate in collaborative filtering based recommender systems algorithms. The main contribution of this paper lies in showing that (1) the presence of gray-sheep users can affect the performance—accuracy and coverage—of collaborative filtering based algorithms, depending on the data sparsity and distribution; (2) gray-sheep users can be identified using clustering algorithms in off-line fashion, where the similarity threshold to isolate these users from the rest of clusters can be found empirically; (3) content-based profile of gray-sheep users can be used for making accurate recommendations. The effectiveness of the proposed algorithm is tested on the MovieLens dataset and community of movie fans in the FilmTrust Website, using mean absolute error, receiver operating characteristic sensitivity, and coverage.**

***Keywords*-Recommender systems; Collaborative filtering; Content-based filtering; Gray-sheep users; Clustering**

I. Introduction

An important task for a recommender system is to ameliorate the *quality* of the recommendations for a user. If a user trusts and leverages a recommender system, and then discovers that they are unable to find what they want then it is unlikely that they will continue with the system. There can be users in recommender systems that have unusual taste as compared to the rest of the community, so collaborative filtering (CF) recommender system would produce poor quality recommendation for these users. Some author have used the term *"gray-sheep"* [4] to distinguish these users from other users. They have low correlation coefficients with other users, as they partially agree or disagree with other users. The presence of these users in a small or medium community of users poses two problems: (1) they may not receive accurate recommendation, even after initial start up phase for users and the system, (2) they may negatively affect the recommendations of the rest of community.

In this paper, we systematically explore gray-sheep user problem. Specifically, we look at four key questions:

1- How can gray-sheep users be detected in a recommender system?
2- Does the presence of gray-sheep users affects the recommendation quality of the community?
3- How does the collaborative filtering algorithms perform over these users?
4- How does the the text categorization algorithms trained on the content profiles of these users perform over these users?

We propose a clustering solution to detect these users in off-line fashion. We propose a *switching hybrid recommender system* [8] for overcoming the aforementioned problems. We show that the proposed approach (1) reduces the recommendation error rate for gray-sheep users, (2) maintains reasonable computational performance. To the best of our knowledge, this is the first attempt to propose a formal solution to satisfy the needs of gray-sheep users. We evaluate our algorithm on the MovieLens[1] and the FilmTrust[2] datasets.

II. Background: Recommender Systems

Recommender system consists of two basic entities: users and items, where users provide their opinions (ratings) about items. We denote these users by $M = \{m_1, m_2, \cdots, m_X\}$, where the number of users using the system is $|M| = X$ and denote the set of items being recommended by $N = \{\ n_1, n_2, \cdots, n_Y\ \}$, with $|N| = Y$. The users will have given ratings of some, but not all of the items. We denote these ratings by $(r_{m_i,n_j}|(m_i, n_j) \in \mathcal{D})$, where $\mathcal{D} \subset M \times N$ denotes the set of user-item pairs that have been rated. We denote the total number of ratings made by $|\mathcal{D}| = T$. Typically each user rates only a small number of the possible items so that $|\mathcal{D}| = T \ll |M \times N| = X \times Y$. It is not unusual in practical systems to have $\frac{T}{(X \times Y)} = 0.01$. We can denote the ratings made by the users by a $X \times Y$ rating matrix R with elements r_{m_i,n_j}. The task of the recommender system is to infer the elements in R for which we do not have any data.

There are two main types of recommender systems: collaborative filtering (CF) and content-based filtering recommender systems. Collaborative filtering based 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. Examples of these systems include the GroupLens system [12], and Ringo[3]. Content-based recommender systems

[1] www.grouplens.org/node/73
[2] http://trust.mindswap.org/FilmTrust/
[3] www.ringo.com

[8] 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 [13], a newsgroup filtering system uses the words of text as features. Hybrid recommender systems have also been proposed, which combine individual recommender systems [8], [7].

III. Related Work

Hybrid recommender systems, emphasizing on the content-based filtering part has been the favourite subject of many research projects like [14], [8], [10], [7]. However, none of them discuss the gray-sheep users problem formally. Generating recommendation using clustering scheme has been proposed in [6], [1]. We use related clustering ideas but in the context of detecting the gray-sheep users and producing scalable recommendations.

The gray-sheep problem was highlighted in [5], where the authors proposed an on-line hybrid recommender system for news recommendation. They used a weighted average approach to combine the collaborative filtering approach with the content-based filtering approach, where the weights are learned per user basis. This approach is very expensive both in terms of memory requirement and time. They did not propose any formal solution focusing specifically on gray-sheep users. In [17], the authors presented the agent-based simulation for distributed knowledge management. They instantiated this framework for collaborative filtering domain using MovieLens dataset. They claimed that gray-sheep agents—agents belonging to several communities—do not affect the regular agents—agents belonging to one community, which is in contrast with our results (see section VI). As it was a simulation, they did not describe a method to identify these users and satisfy their needs. Furthermore, both of these approaches do not provide any benchmarks, making it unclear how to compare the proposed algorithm with them.

IV. Dataset and Evaluation Metrics

A. Dataset

We have used the MovieLens (ML) and FilmTrust (FT) datasets for evaluating our algorithm. The MovieLens dataset contains 943 users, 1682 movies, and $100\,000$ ratings on an integer scale of 1 (bad) to 5 (excellent). We created the second dataset by crawling the FilmTrust website. The dataset retrieved (on 10th of March 2009) contains 1214 users, 1922 movies, and $28\,645$ ratings on a floating point scale of 1 (bad) to 10 (excellent). The sparsity of the datasets is found to be 0.934 and 0.988 for ML and FT dataset respectively. The sparsity of a dataset is calculated as follows: $\left(1 - \frac{\textit{non zero entries}}{\textit{all possible entries}}\right)$.

B. Feature extraction and selection

We downloaded information about each movie in the MovieLens and FilmTrust dataset from the IMDB[4]. After stop word[5] removal and stemming[6] we constructed a feature vector of keywords, tags, genres, directors, actors/actresses, producers, writers, and user reviews given to a movie in the IMDB. We used *TF-IDF* (Term Frequency-Inverse Document Frequency) approach for determining the weights of words (features) in a movie. The details of training the classification and regression approaches using these features can be found in our previous work [8].

C. Metrics

Our specific task in this paper is to predict scores for items that have already been rated by actual users, and to check how well this prediction helps users in selecting high quality items. Considering this, we have used *Mean Absolute Error (MAE)*, *Receiver Operating Characteristic (ROC) sensitivity* and coverage [3], [16], [10], [7], [9]. We calculated these metrics for each user and reported the average results.

V. Proposed Algorithm

A. Detecting gray-sheep users: A clustering solution

This concept has been inspired from the shape detection in image processing domain. While generating clusters for shape detection, a separate cluster is made for the features that are not very similar with the current clusters. Other researchers have found that this can result in decrease of error [15] in shape detection. Gray-sheep users can be handled in the same way, and can be separated into a distinct cluster using our proposed algorithm, Algorithm 1.

We modified the K-Means++ clustering algorithm [2] for clustering the user-item rating matrix and detecting gray-sheep users. K-Means++ uses a probabilistic approach for choosing the centroids and claims that it yields much finer clusters than the K-Means clustering algorithm, resulting in a solution that is $O(\log k)$ competitive to the optimal k-means solution. We used the K-Means++ concept over the so-called *power users*—users that have rated a large number of items in a recommender system [11]. Let m_p denotes the user who has rated the maximum number of items. The probability of a user m_i to be a power user is calculated as follows:

$$P(m_i) = \frac{I_{m_i}}{I_{m_p}}\,, \tag{1}$$

where I_{m_i} and I_{m_p} denote the number of items rated by user m_i and m_p respectively. The pseudo-code of the proposed clustering algorithm is outlined in Algorithm 1.

The steps from 4 to 7 are essentially the same as in K-Means clustering algorithm [2]. In step 22 (within procedure CENTROIDSELECTION), the $\log$ is used for minimizing the effect of larger values of $\frac{1}{P(i)}$. This equation aims at finding the centroid with the farthest distance from the current chosen centroid (i.e. with minimum Pearson correlation), and with the greatest probability for being a power user (i.e. with maximum value of $P(i)$ shown in equation 1).

[4] www.imdb.com

[5] We used Google's stop word list www.ranks.nl/resources/stopwords.html.

[6] We used Porter Stemmer algorithm for stemming.

The proposed algorithm has advantages over the traditional K-Means clustering algorithm and is chosen because: (1) it gives coherent size and homogeneous clusters, which are essential to detect gray-sheep users, (2) the convergence is faster.

The steps 8 to 15 detect gray-sheep users by grouping users having similarity (with other clusters) less than a pre-defined threshold sim_{thr} into a separate cluster.

Algorithm 1 : Clusters the user-item rating matrix into $k+1$ clusters, and groups gray-sheep users into a separate cluster

1: **procedure** PROBKMEANS++(M, k, MAX, sim_{thr})
2: $C = \{c_1, c_2, \cdots c_k\}$ = CENTROIDSELECTION (k)
3: $t \leftarrow 0$
4: **repeat**
5: $t \leftarrow t + 1$
6: Set the cluster D_i, for each $i \in 1, ..., k$, to be the set of points in M that are closer to c_i than they are to c_l for all $l \neq i$.
7: Set c_i, for each $i \in 1, ..., k$, to be the center of mass of all points in D_i where,

$$c_i = \frac{1}{|D_i|} \sum_{m \in D_i} m \,.$$

8: **if** $t = MAX$ **then**
9: Create a new Centroid c_{k+1}
10: **for all** $m \in M$ **do**
11: **if** $d(m) < sim_{thr}$ **then**
12: assign point m to cluster D_{k+1}
13: **end if**
14: **end for**
15: **end if**
16: **until** ($t = MAX$)
17: **return** (C, D)
18: **end procedure**

19: **procedure** CENTROIDSELECTION(M, k)
20: **repeat**
21: Choose the initial centroid c_1 to be m_p.
22: Choose the next centroid c_i by selecting $c_i = m' \in M$ with probability:

$$Prob = \arg\min \left(\frac{d(m')}{\sum_{m=1}^{M} d(m)^2} + \log\left(\frac{1}{P(m')} + 1\right) \right).$$

▷ $d(m)$ represents the shortest distance from a data point m to the closest centroids we have already chosen
23: **until** k centroids are found
24: **return** $\{ c_1, c_2, \cdots c_k \}$ ▷ k centroids,
25: **end procedure**

B. Generating recommendations for gray-sheep users

The recommendations for gray-sheep users are generated using the cluster based CF algorithm as proposed in [1]. We assume that the CF based algorithms would not perform well over these users, because they have partial agreement with the rest of the community. This assumption argues that the recommendations can be generated based on the content profile of these users (by training the machine learning classifiers) and ignoring the contributions of the community (neighbours).

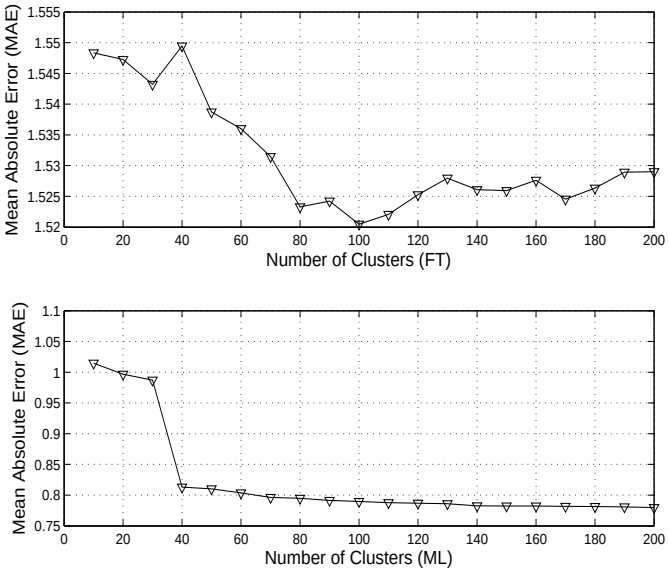

Fig. 1. Finding the optimal number of clusters for MovieLens (ML) and FilmTrust (FT) datasets through training set.

We trained (using the Weka library[7]) the following text categorization algorithms over the content profiles of these users: KNN, Naive Bayes classifier (NB), Decision Tree (C4.5), Support Vector Machines classification (SVMClass), and Support Vector Machines regression (SVM Reg). Furthermore, we tunned them for the optimal parameters on the training set. For the SVM regression, we used nu-SVR version of the SVM regression using the LibSVM[8] library. We used the linear kernel and trained the cost parameter.

C. Generating recommendation for the remaining users using the CF

The recommendations for the users not identified as gray-sheep users are generated using the cluster based collaborative filtering [1]. For measuring the similarity between an active user and the cluster centroid, we used the Pearson correlation.

VI. RESULTS AND DISCUSSION

A. Finding the optimal system parameters

1) Optimal number of clusters: For finding the optimal number of clusters, we changed the cluster size from 10 to 200 with a difference of 10 (keeping the remaining parameters fixed), and measured the corresponding MAE, over the training set. Fig. 1 shows that the MAE decreases with an increase in the number of clusters. For ML dataset the MAE keeps on decreasing, however after 140 clusters the decrease is very

[7] www.cs.waikato.ac.nz/ml/weka/
[8] http://www.csie.ntu.edu.tw/ cjlin/libsvm

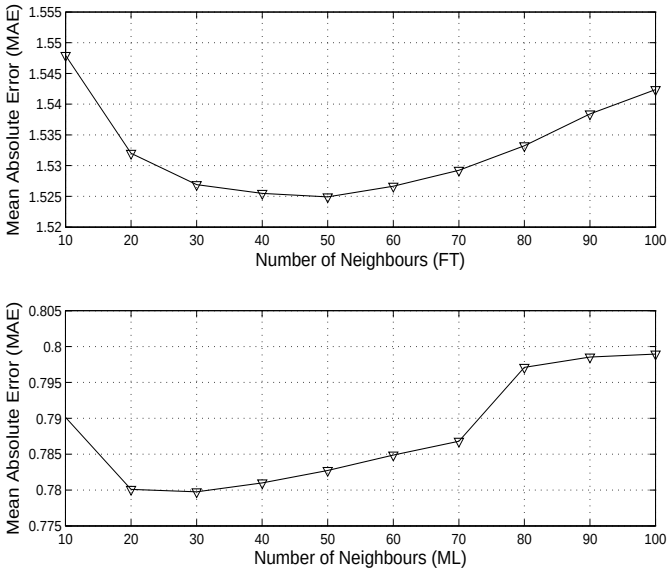

Fig. 2. Finding the optimal number of neighbours for MovieLens (ML) and FilmTrust (FT) datasets through training set.

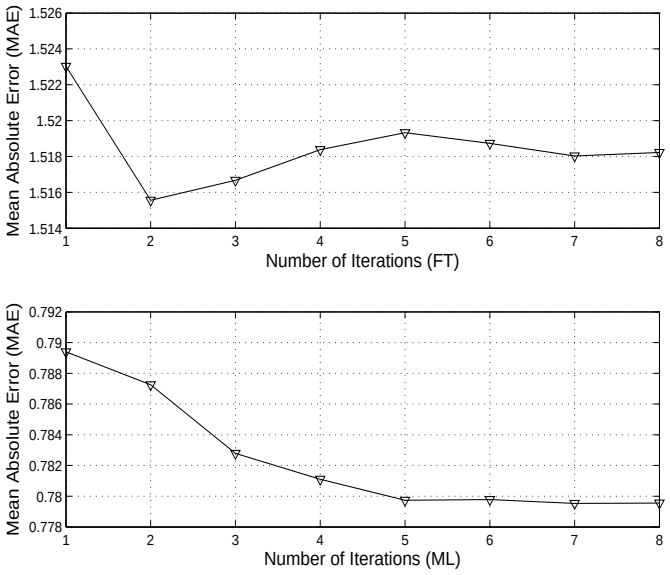

Fig. 3. Finding the optimal number of iterations for MovieLens (ML) and FilmTrust (FT) datasets through training set.

insignificant. For FT dataset, the MAE is minimum for 100 clusters, and then increases with an increase in the number of clusters. For this reason we choose the cluster size to be 140 and 100 for ML and FT datasets respectively for subsequent experiments.

2) Optimal number of neighbours for collaborative filtering: We changed the number of neighbours from 10 to 100 and measured the corresponding MAE. Fig. 2 shows how the MAE changes as a function of neighbourhood size. We note that in the case of ML dataset, the MAE keeps on decreasing with an increase in the number of neighbours, reaches at its minimum for neighbourhood size of 50, and then starts increasing again.

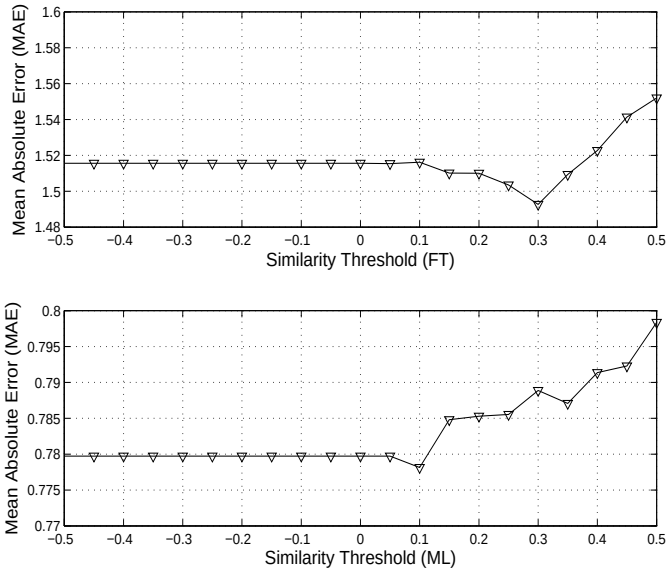

Fig. 4. Finding the optimal similarity threshold for MovieLens (ML) and FilmTrust (FT) datasets through training set.

For the FT dataset, the neighbourhood size of 20 gives the lowest MAE. We choose the optimal neighbourhood sizes to be 50 and 20 for the ML and FT datasets respectively.

3) Optimal number of iterations: Fig. 3 shows how the MAE changes with an increase in the number of iterations. For ML dataset the MAE keeps on decreasing with an increase in the number of iterations until it converges. After 5 iterations, the difference in MAE observed between two iterations becomes very small. To keep a good balance between computation and performance requirement, we choose the optimal number of iterations to be 5. Similarly, we tunned the optimal number of iterations for FT dataset, which are found to be 2.

4) Optimal similarity threshold to detect gray-sheep users: The similarity threshold sim_{thr} parameter determines and controls the number of gray-sheep users. A large value of sim_{thr} (e.g. $sim_{thr} = 1$) would results in all users being gray-sheep users, whereas a smaller value of sim_{thr} (e.g. $sim_{thr} = -1$) would result in no user being gray-sheep user. We changed the value of sim_{thr} from 1.0 to -1.0 with a difference of 0.05 and measured the corresponding MAE of the users not identified as gray-sheep users. The value of sim_{thr} that gives the minimum MAE, is termed as the optimal threshold value of sim_{thr}. Fig. 4 shows how the MAE changes with a change in sim_{thr}. We observe that the MAE is minimum at $sim_{thr} = 0.1$ and $sim_{thr} = 0.3$ for ML and FT datasets respectively. A further increase in the similarity threshold increases the MAE. We chose these optimal values for the subsequent experiments. Hence the answer to the question: *"How can gray-sheep users be detected in a recommender system?"* is, these users can be detected using a clustering algorithm, where the similarity threshold to isolate these users from the rest of clusters can be found empirically.

B. Results of CF based algorithms

We show the results of the cluster based KNN algorithm over (1) all users (2) gray-sheep users[9] (3) the users not classified as gray-sheep users. Taking the results of table I into account, the answer to the question *"does the presence of these users make any difference on the recommendations of community as a whole?"* is that, it is dataset dependent. Their presence does not make any difference on the recommendation quality, in the case of MovieLens dataset, because only 18 users are detected as gray-sheep users. It does make some difference in the case of FilmTrust dataset (with 1.33% improvement in MAE), as greater number of users are detected as gray-sheep users (157). Considering these results, we claim that the presence of a large number of gray-sheep users may significantly affect the recommendations quality of the community.

We observe that the performance of cluster based collaborative filtering suffers the most for gray-sheep users, because they rely on the similar users (neighbours). In this case, the correlation coefficient is poorly approximated, and thus less reliable recommendations are produced. The $\%$ increase in the MAE for gray sheep users (as compared to all users) is 2.2% for the MovieLens and 9.06% for the FilmTrust dataset. Keeping these results into account, the answer to the question: *"what is the performance of collaborative filtering based recommendation algorithms for these users?"* is that, the collaborative filtering fails to produce good recommendation for these users.

C. Results of text categorization based algorithms for gray-sheep users

To answer the question: *"how text categorization algorithms trained on the content profile of users perform over these users?"*, we perform experiments with the algorithms discussed in V-B. The result have been shown in table II. The result shows that these algorithms improve the recommendation quality. We note that the SVM regression outperforms the rest, with 5.39% and 11.35% improvement over the cluster based collaborative filtering's result in the case of MovieLens and FilmTrust dataset respectively. We further note that, the SVM regression gives much better results over the FilmTrust as compared to the MovieLens dataset. This is because the FilmTrust dataset is well suited to regression algorithms (see section IV-A).

D. A comparison of different algorithms

Table III shows how different algorithms perform over the MovieLens and FilmTrust datasets. We observe that, for the MovieLens dataset, the CF based algorithm performs the best, whereas for FilmTrust dataset, the SVM regression perform the best. It is because, FilmTrust dataset is relatively sparse as compared to MovieLens dataset[10].

[9]After detecting the gray-sheep users, the steps of algorithm 1 from 8 to 15 are not executed. This ensures that a user groups with the most similar clusters.

[10]In the FilmTrust dataset, the rating profiles of users lead to poorly approximated (Pearson) correlation coefficient resulting in less reliable predictions.

We observe from table II and III, that CF based algorithm gives good result for MovieLens and FilmTrust dataset, however the performance degrades for gray-sheep users. The reason is that gray-sheep users have unclear rating profile, and in the worse case, we might find very few or no similar users (neighbours) for a gray-sheep user. The text categorization approaches give good results, for these users, as they make effective use of users content profile that are used for making predictions.

E. Complexity of the proposed algorithm

The training of SVM can be done off-line and it has time complexity of $O(Y^3)$ and training memory span of $O(Y^2)$. To make a prediction, SVM takes $O(n_{sv})$, where n_{sv} is the number of support vectors. In our case, we have G gray-sheep users, hence the off-line and on-line time complexity will be $O(GY^3)$ and $O(n_{sv})$ respectively.

The training cost of the proposed clustering algorithm is $O(XY)$[11]. To make a prediction for a user using the clustering technique, we find the active user's similarity with k other clusters, which takes $O(k)$ calculations. We do it for $X-G$ users (where G is the number of gray-sheep users), hence the on-line time complexity becomes $O(k(X-G))$.

The off-line and on-line cost of the proposed algorithm is $O(GY^3)+O(XY)$ and $O(n_{sv})+O(k(X-G))$ respectively. It must be noted that the on-line cost is less than conventional collaborative filtering based algorithms [3], [16] and any conventional hybrid algorithm, for example [5].

VII. Conclusion

In this paper, we pointed out the gray-sheep users problem responsible for the increased error rate in collaborative filtering based recommender systems algorithms. A clustering solution is proposed to detect these users and the recommendations for these users are generated based on the SVM regression trained on the content profiles of the users, whereas for other users based on the cluster based collaborative filtering.

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 Board's Collaborative Research and Development programme.

References

[1] S.K.L. Al Mamunur Rashid, G. Karypis, and J. Riedl, *ClustKNN: a highly scalable hybrid model-& memory-based CF algorithm*, Proc. of WebKDD 2006: KDD Workshop on Web Mining and Web Usage Analysis, August 20-23 2006, Philadelphia, PA, Citeseer.

[2] D. Arthur and S. Vassilvitskii, *k-means++: The advantages of careful seeding*, Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics Philadelphia, PA, USA, 2007, pp. 1027–1035.

[11]In addition to this, we incur the cost of choosing the cluster centroids, which is $O(KX)$.

TABLE I

THE PERFORMANCE OF THE CLUSTER BASED CF ALGORITHM FOR DIFFERENT TYPES OF USERS. "ALL" REPRESENTS ALL USERS, "GS" REPRESENTS GRAY-SHEEP USERS, AND "REMAINING" REPRESENTS THE USERS NOT IDENTIFIED AS GRAY-SHEEP.

Dataset	MAE			Coverage		
	All	**GS**	**Remaining**	**All**	**GS**	**Remaining**
ML	0.7736 ± 0.0027	0.7908 ± 0.1602	0.7730 ± 0.0027	99.8190	99.9283	99.8186
FT	1.4803 ± 0.0056	1.6145 ± 0.0541	1.4606 ± 0.0058	95.5699	91.4841	95.6704

TABLE II

THE PERFORMANCE IN TERMS OF MAE, ROC-SENSITIVITY, AND COVERAGE OF DIFFERENT TEXT CATEGORIZATION ALGORITHMS COMPUTED OVER GRAY-SHEEP USERS.

Dataset	Approach	MAE	ROC-Sensitivity	Coverage
ML	SVM Reg	$\mathbf{0.7481 \pm 0.1214}$	0.6689 ± 0.1218	100
	SVMCalss	0.7701 ± 0.1547	$\mathbf{0.7042 \pm 0.2021}$	100
	NB	0.8122 ± 0.1671	0.6981 ± 0.2020	100
	KNN	0.7872 ± 0.1558	0.6460 ± 0.2126	100
	C4.5	0.8180 ± 0.1522	0.6312 ± 0.2021	100
	CCF	0.7908 ± 0.1602	0.7012 ± 0.0397	99.8190
FT	SVM Reg	$\mathbf{1.4312 \pm 0.0601}$	$\mathbf{0.5651 \pm 0.0487}$	100
	SVMCalss	1.4591 ± 0.0612	0.5260 ± 0.0468	100
	NB	1.4724 ± 0.0598	0.5187 ± 0.0468	100
	KNN	1.4821 ± 0.0524	0.5089 ± 0.0478	100
	C4.5	1.4980 ± 0.0578	0.5035 ± 0.0452	100
	CCF	1.6145 ± 0.0542	0.4822 ± 0.0442	91.4841

TABLE III

THE PERFORMANCE IN TERMS OF MAE, ROC-SENSITIVITY, AND COVERAGE OF DIFFERENT TEXT CATEGORIZATION ALGORITHMS COMPUTED OVER ALL USERS.

Dataset	Approach	MAE	ROC-Sensitivity	Coverage
ML	SVM Reg	0.7932 ± 0.0038	0.6589 ± 0.0055	100
	SVMCalss	0.8101 ± 0.0070	0.685 ± 0.0037	100
	NB	0.8250 ± 0.0105	0.6871 ± 0.0049	100
	KNN	0.8320 ± 0.0132	0.6790 ± 0.0082	100
	C4.5	0.8480 ± 0.1522	0.6312 ± 0.0110	100
	CCF	$\mathbf{0.7736 \pm 0.0041}$	$\mathbf{0.7371 \pm 0.0094}$	99.9283
FT	SVM Reg	$\mathbf{1.4492 \pm 0.0191}$	$\mathbf{0.5421 \pm 0.0087}$	100
	SVMCalss	1.4631 ± 0.0212	0.5120 ± 0.0128	100
	NB	1.4812 ± 0.0298	0.5152 ± 0.0110	100
	KNN	1.511 ± 0.0224	0.4912 ± 0.0121	100
	C4.5	1.5210 ± 0.0278	0.4887 ± 0.0210	100
	CCF	1.4606 ± 0.0241	0.5101 ± 0.0175	95.5699

[3] John S. Breese, David Heckerman, and Carl Kadie, *Empirical analysis of predictive algorithms for collaborative filtering*, Morgan Kaufmann, 1998, pp. 43–52.

[4] Robin Burke, *Hybrid recommender systems: Survey and experiments*, User Modeling and User-Adapted Interaction **12** (2002), no. 4, 331–370.

[5] Mark Claypool, Anuja Gokhale, Tim Mir, Pavel Murnikov, Dmitry Netes, and Matthew Sartin, *Combining content-based and collaborative filters in an online newspaper*, In Proceedings of ACM SIGIR Workshop on Recommender Systems, 1999.

[6] Mark Connor and Jon Herlocker, *Clustering items for collaborative filtering*, 2001.

[7] M.A. Ghazanfar and A. Prugel-Bennett, *An Improved Switching Hybrid Recommender System Using Naive Bayes Classifier and Collaborative Filtering*, Lecture Notes in Engineering and Computer Science: Proceedings of The International Multi Conference of Engineers and Computer Scientists 2010, IMECS 2010, 17–19 March, 2010, Hong Kong, 2010, pp. 493–502.

[8] ———, *Building Switching Hybrid Recommender System Using Machine Learning Classifiers and Collaborative Filtering*, IAENG International Journal of Computer Science **37** (2010), no. 3, 272–287.

[9] ———, *Novel Significance Weighting Schemes for Collaborative Filtering: Generating Improved Recommendations in Sparse Environments*, DMIN'10, the 2010 International Conference on Data Mining, WORLDCOMP'10, 12–15 July, 2010, USA, 2010.

[10] ———, *A scalable, accurate hybrid recommender system*, The 3rd International Conference on Knowledge Discovery and Data Mining (WKDD 2010), IEEE, 9–10 January, 2010, Thailand, 2010.

[11] Loren G. Terveen Jonathan L. Herlocker, Joseph A. Konstan and John T. Riedl, *Evaluating collaborative filtering recommender systems*, ACM Transactions on Information Systems (TOIS) archive **22** (2004), 734–749.

[12] Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, and John Riedl, *Grouplens: applying collaborative filtering to usenet news*, Commun. ACM **40** (1997), no. 3, 77–87.

[13] K. Lang, *Newsweeder: Learning to filter netnews*, In Proceedings of the Twelfth International Conference on Machine Learning, 1995.

[14] Michael J. Pazzani, *A framework for collaborative, content-based and demographic filtering*, Artificial Intelligence Review **13** (1999), no. 5 - 6, 393–408.

[15] A. Ramanan and M. Niranjan, *Resource-Allocating Codebook for Patch-based Face Recognition*, 2009 International Conference on Industrial and Information Systems (ICIIS 2009), 2009, pp. 268–71.

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

[17] M. Wurst, *Evaluating Knowledge Management in Heterogeneous Domains by Agent-based Simulation*, Agent Mediated Knowledge Management, 82.