Evaluation of a Hybrid Recommender System

Soumyanil Banerjee
EECS Department
soumbane@umich.edu

Kevin Fries
Department of Civil and Environmental Engineering
kjfries@umich.edu

Gaurav Kumar Singh
EECS Department
gauravs@umich.edu

Ajay Vasudevan
EECS Department
ajayvasu@umich.edu

Abstract

Recommender Systems are systems that filter information for users. In this project, we have applied the fuzzy K-means and adjusted K-means clustering technique to generate item feature based similarity graphs. Outputs of the item based clustering approaches are then combined with user-rating based similarity graphs and fed into a collaborative filtering framework to predict user movie ratings and thereby make recommendations for new movies. This hybrid analysis is performed using a methodology outlined by Li and Kim [1]. This hybrid approach performed better than a collaborative filtering approach alone.

1 Introduction

Recommender systems are utilized on several websites, such as Netflix, Amazon, YouTube, and Twitter, to help users identify new content that they might like. These recommender systems help increase usage of the service in question, but poor recommenders may deter users as recommendations do not match their tastes. Because recommender systems are so useful, a number of algorithms have been developed, each with differing advantages. These algorithms generally fall under two categories: collaborative filtering and content-based filtering.

Collaborative filtering uses information about a users tastes (such as movie rating, page-views, or favorited tweets) to try and determine whether or not that user will like new or unseen content. This has the advantage of not having to actually know anything about the content in question. Content-based filtering approaches utilize a series of discrete characteristics of an item (e.g. genre, quality, length, etc.) in order to recommend additional items with similar features to ones users tend to view. From a mathematical perspective, recommender systems are a problem in matrix completion. Given a sparse matrix of users by items, what is our guess at what a zero entry would be if the user went and rated that item.

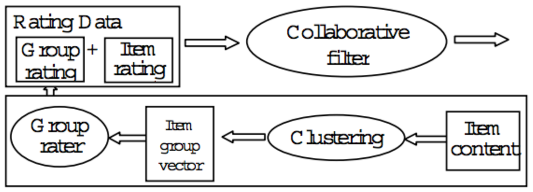

Figure 1: Overview of Hybrid Recommender Approach [1]

In this work, we replicate the approach used by Li and Kim (2003) [1] in which a hybridization of these approaches is used. First, a content-based algorithm clusters the items of interest into groups by either adjusted k-means or fuzzy k-means. Then it computes the sub-similarity matrix of the clusters. Next, a sub-similarity matrix is generated based on user ratings. The final similarity matrix is a linear combination of the sub-similarity matrices. Finally, this similarity matrix is fed into a collaborative filter to yield recommendations for a user. This process is shown in Figure 1. We then compare this approach to a low rank matrix factorization technique as a point of comparison.

2 Related Work

Initially most recommender systems were simple query-based Information Retrieval Systems or content-based based recommender system which recommended web pages based on content similar to user queries [3]. The first application of collaborative filtering technology to recommender systems was done by Goldberg [4] where the author used manual annotations to identify similar users. The first automated predictions for collaborative recommender systems were done independently by Grouplens [5] and Ringo [6].

The evolution of recommender systems has shown the importance of hybrid techniques, which merge different aspects of varying algorithms in order to take advantage of the benefits of each method. A survey focused on hybrid recommender systems has been presented in [7]. The neighborhood-based collaborative filter was the recommendation method most popular at the beginning of recommender system implementations; Herlocker et al. [8] provides a set of guidelines for designing neighborhood-based prediction systems. Adomavicius and Tuzhilin [9] present an overview on the recommender system field, pointing out the most complex areas in which researchers in recommender systems should focus in the next generation of recommender systems: limited content analysis and overspecialization in content-based methods, cold-start and sparsity in collaborative filter methods, model-based techniques, non-intrusiveness, flexibility (real-time customization), and more.

Recommender systems are currently based on demographics, content/attributes, and collaborative filtering. However, currently, these systems are beginning to incorporate social information. In the future, they will use implicit, local and personal information from the Internet of Things [9].

3 Methods

3.1 Data

The data used for this experiment is the MovieLens 100k dataset, which has 100,000 user ratings from 943 users on 1682 movies [5]. Each user has rated at least 20 movies, and each movie is assigned binary labels for 19 different movie genres (each movie can have more than one genre) as well as a release date.

3.2 Hybrid Recommender System

The Hybrid Recommender System can be defined in four steps. First, a user-rating based similarity graph is generated. Second, items are clustered based on features, with the cluster assignments being used to generate a second similarity graph. Third, these two similarity graphs are linearly combined. Finally, this new similarity graph is used in a collaborative filter to generate rating predictions, and the highest predictions are recommended to the user.

To generate the user-based similarity graph, we use the Pearson correlation-based similarity given by:

$$sim(k,l) = \frac{\sum_{u=1}^{m}(R_{u,k}-\bar{R}_k)(R_{u,l}-\bar{R}_l)}{\sqrt{\sum_{u=1}^{m}(R_{u,k}-\bar{R}_k)^2}\sqrt{\sum_{u=1}^{m}(R_{u,l}-\bar{R}_l)^2} \tag{1}$$

where $sim(k,l)$ is the similarity between item k and item l, $R_{u,k}$ is the rating by user u on item k, and $\bar{R}_k$ is the average rating of item k by all users. Each movie will then be assigned a similarity between -1 and 1. It is important to note here that only those users who have rated both items k and l are included in the calculation. If no user has rated both items, then a similarity of 0 is assigned.

Next, we cluster the items based on their features. We want soft cluster boundaries because most movies fall under multiple different genres (e.g. Scream would be considered both a comedy and a horror film) and users have tendencies to like broad ranging types of movies. To achieve this end, we use both an adjusted k-means algorithm and a fuzzy k-means algorithm.

In fuzzy k-means, we alter the cost function in k-means to include a membership term:

$$\arg\min_{C} \sum_{i=1}^{n} \sum_{j=1}^{c} w_{i,j}^{m} ||x_i - c_j||^2 \tag{2}$$

where

$$w_{i,j}^{m} = \frac{1}{\sum_{k=1}^{c} \left(\frac{||x_i - c_j||}{||x_i - c_k||}\right)^{\frac{2}{m-1}}} \tag{3}$$

Here, $w_{i,j}^{m}$ is the membership weight, x_i is the feature vector for item i, c_j is the center for cluster j, and m is the "fuzziness" of the cluster. As m approaches 1, the cluster weights approach 0 or 1, indicating hard cluster assignments.

In adjusted k-means, instead of altering the cost function such that membership weights are assigned during the optimization, we instead assign membership after performing standard k-means. The membership weight is defined by:

$$w_{i,j} = \frac{\left(\frac{1}{dis_{i,j}}\right)^{\frac{2}{m-1}}}{\sum_{r=1}^{c} \left(\frac{1}{dis_{r,j}}\right)^{\frac{2}{m-1}}} \tag{4}$$

where $dis_{i,j}$ is the Euclidean Distance between item j and cluster center r, and m is the same "fuzziness" factor as before.

After implementing either clustering algorithm, we are left with a membership matrix U with dimension of the number of items by the number of clusters. We can use this matrix to generate a second similarity graph via the adjusted cosine similarity:

$$sim(k,l) = \frac{\sum_{c=1}^{m} (U_{c,k} - \bar{U}_c)(U_{c,l} - \bar{U}_c)}{\sqrt{\sum_{u=1}^{m} (U_{c,k} - \bar{U}_c)^2} \sqrt{\sum_{c=1}^{m} (U_{c,l} - \bar{U}_c)^2}} \tag{5}$$

where $sim(k,l)$ is the similarity between item k and item l, $U_{c,k}$ is the membership in cluster c for item k and $\bar{U}_c$ is the average membership of all items in cluster c. This is similar to the Pearson similarity, except we replace the average over each item ($\bar{R}_k$) with the average over each cluster ($\bar{U}_c$). Unlike in the user-rating based method, all items will have some membership in all clusters. Therefore, no exclusion of items or clusters is necessary.

Finally, the two similarity graphs are linearly combined to get a hybrid similarity graph. This is done by:

$$sim_{total}(k,l) = sim_{rating}(k,l) * (1-c) + sim_{item}(k,l) * c \tag{6}$$

Given this final similarity graph, it is possible to feed this information into a CF to determine a user's rating of an unrated item:

$$P_{u,k} = \bar{R}_k + \frac{\sum_{i=1}^{n} (R_{u,i} - \bar{R}_i) * sim_{total}(k,i)}{\sum_{i=1}^{n} |sim_{total}(k,i)|} \tag{7}$$

where $P_{u,k}$ is the predicted rating of item k for user u, n is the number of nearest neighbors you would like to compare to, and other variables are defined as before. In this approach, again it is important to note that if the user has not rated neighbor n, it is not included in the summation.

3.3 Low Rank Collaborative Filtering Algorithm

Low Rank collaborative filtering is a form of sparse matrix completion. The collaborative filtering algorithm in the setting of movie recommendations considers a set of n-dimensional parameter vectors $x^{(1)}, ..., x^{(n_m)}$ and $\theta^{(1)}, ..., \theta^{(n_u)}$, where the model predicts the rating for movie i by user j as $y^{(i,j)} = (\theta^{(j)})^T x^{(i)}$. Given a dataset that consists of a set of ratings produced by some users on

some movies, we wish to learn the parameter vectors $x^{(1)}, ..., x^{(n_m)}$ and $\theta^{(1)}, ..., \theta^{(n_u)}$ that produce the best fit (minimizes the squared error).
The collaborative filtering cost function including regularization is given by 8

$$J(x^{(1)}, ..., x^{(n_m)}, \theta^{(1)}, ..., \theta^{(n_u)}) = \frac{1}{2} \sum_{i,j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + (\frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} (\theta_k^{(j)})^2) + (\frac{\lambda}{2} \sum_{j=1}^{n_m} \sum_{k=1}^{n} (x_k^{(i)})^2) \tag{8}$$

Whenever a new user appears, the algorithm requires the user to rate certain items so as to match it to another user with the similar taste. After the additional ratings have been added to the dataset, the collaborative filtering model is trained. The trained model learns the parameters x and θ. To predict the rating of movie i for user j, $(\theta^{(j)})^T x^{(i)}$ is computed, which essentially is an estimate of all the ratings by all the users. The top rated items are recommended to the user.

3.4 Toy Problem

Table 1: Item Rating

	User1	**User2**	**User3**
Independence Day	3	4	
Mars Attacks!	3	4	4
Men in Black	4	5	
Sharknado	1	1	

Table 2: Group Rating

	Cluster1	**Cluster2**
Independence Day	40%	57%
Mars Attacks!	40%	57%
Men in Black	45%	55%
Sharknado	50%	50%

Here we present a toy problem to assist in understanding of the implementation of the algorithm.

1. Using the movie features, we apply fuzzy clustering methods in order to form groups of movies (see Table 2) where for each movie, we get the membership probability for the item belonging to each cluster.
2. After clustering, we calculate the similarity measures. Our final similarity measure is a linear combination of the user ratings based Pearson Correlation Similarity and the clustering based Adjusted Cosine similarity. The linear combination is calculated according to Equation 6 and the best combination coefficient can be chosen by model selection.

   For example, to calculate the similarity between Independence Day and Sharknado, we first follow Equation 1 for item based similarity and Equation 5 for clustering based similarity. It is important to restate here that when computing average ratings and similarities between two movies, you *only* look at users that have rated both movies. Otherwise, zero ratings will skew the similarity scores.

$$sim(I, S)_{Pearson} = \frac{(3 - 3.5) \times (1 - 1) + (4 - 3.5) \times (1 - 1)}{\sqrt{(3 - 3.5)^2 + (4 - 3.5)^2}\sqrt{(1 - 1)^2 + (1 - 1)^2}} \tag{9}$$

$$sim(I,S)_{Cosine} = \frac{(0.4-0.44)\times(0.5-0.44)+(0.57-0.55)\times(0.5-0.55)}{\sqrt{(0.4-0.44)^2+(0.57-0.55)^2}\sqrt{(0.5-0.44)^2+(0.5-0.55)^2}} \tag{10}$$

And when the Combination Coefficient is 0.4

$$Sim(G,S)_{LinearCombination} = 0 \times (1-0.4) + (-0.97) \times 0.4 = -0.584 \tag{11}$$

3. To predict ratings we use Equation 7 and can predict that User 3 would rate Independence Day as a 4.

4 Results

4.1 Performance Evaluation - Mean Absolute Error (MAE)

Mean Absolute Error (MAE) is commonly used in evaluating the accuracy of a recommender system by comparing the numerical recommendation scores against the actual user ratings in the test data. The MAE is calculated by summing the absolute errors of the corresponding rating-prediction pairs and then normalizing by the number of pairs.

4.2 Clustering

As outlined above, we assess two different clustering methodologies: adjusted k-means and fuzzy k-means. We compare how each of these methods improve the rating predictions, and we find that in general neither shows real advantage over the other. However, fuzzy k-means is more computationally complex (see Figure 2, so when dealing with larger datasets, it would be best to use adjusted k-means to reduce computation cost.

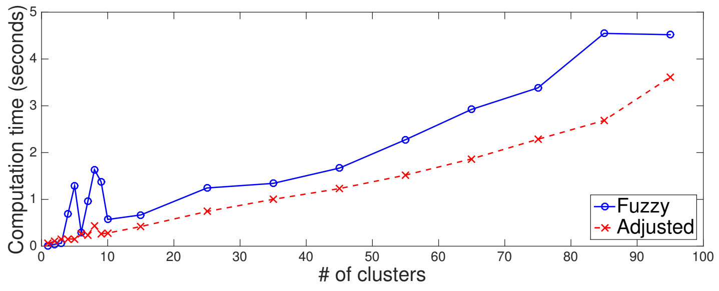

Figure 2: Number of Clusters vs Computation time for Fuzzy K-means and Adjusted K-means Clustering ($m = 1.45$, $c = 0.2$, $n = 20$, number of movies = 1682)

4.3 Number of Clusters vs MAE

Experiment results (see Figure 3) suggest that the number of clusters (k) impacts the quality of prediction. Immediately we see a sharp downward spike when the number of clusters is 2, with relatively constant results for anything greater than 5 or 10 (depending on the clustering method). This is in sharp contrast to the findings of [1], who found that 30 clusters yielded the greatest improvement. However, the MAE in their case was on the order of 0.75 for any number of clusters between 5 and 100, and they did not test any values of k less than 3, resulting in an inability to capture the results we show here. We believe 2 clusters resulted in the best assignment because each of the features is a binary label. An alternative clustering method, such as the use of a decision tree (see Trochesset & Bonner[2]), may have yielded better results with binary labeled data.

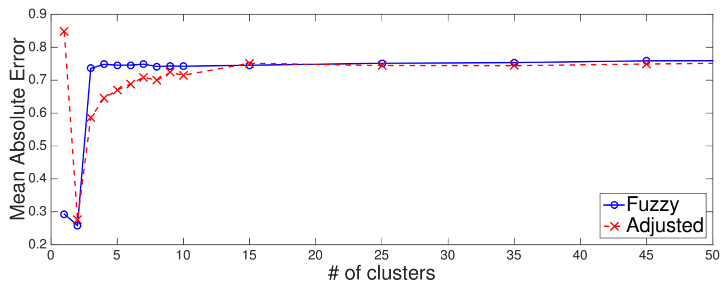

Figure 3: Number of Clusters vs Mean Absolute Error for Fuzzy K-means and Adjusted K-means Clustering ($m = 1.45$, $c = 0.2$, $n = 20$)

4.4 Linear Combination Coefficient vs MAE

In order to find the optimal linear combination coefficient c in Equation 5, we conduct a series of experiments by changing the coefficient from 0 to 1 with a constant step 0.1. Figure 4 shows that when the coefficient arrives at 0.2, an optimal recommendation performance is achieved for each clustering method. This again differs from the results of Li & Kim. This can be attributed to using fewer clusters. If we instead run the experiment using a k value of 30 (see Figure 5), as was optimal for Li & Kim, we find that we can replicate their results.

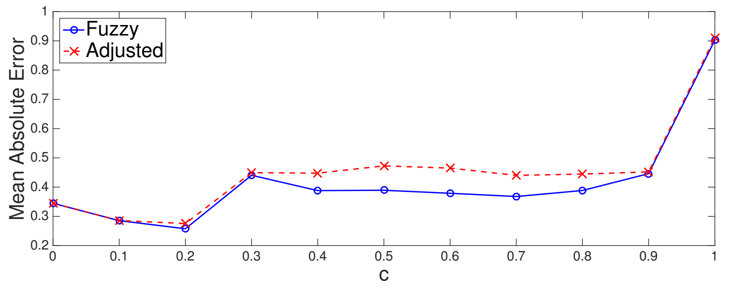

Figure 4: Coefficient of Linear Combination of similarity measures vs Mean Absolute Error for Fuzzy K-means and Adjusted K-means Clustering ($k = 2$, $m = 1.45$, $n = 21$)

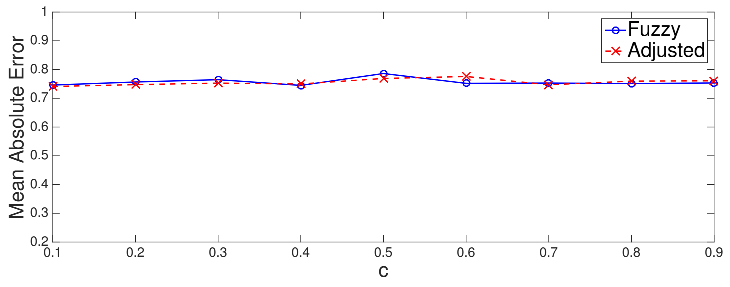

Figure 5: Coefficient of Linear Combination of similarity measures vs Mean Absolute Error for Fuzzy K-means and Adjusted K-means Clustering ($k = 30$, $m = 1.45$, $n = 21$)

4.5 Number of Nearest Neighbors vs MAE

As observed from Figure 6, the size of neighborhood (n) also impacts performance. Our best performance occurs around 20, though the MAE plateaus after 10 with a spike at 15 that we cannot explain. This spike is there no matter how the training and testing sets are segmented.

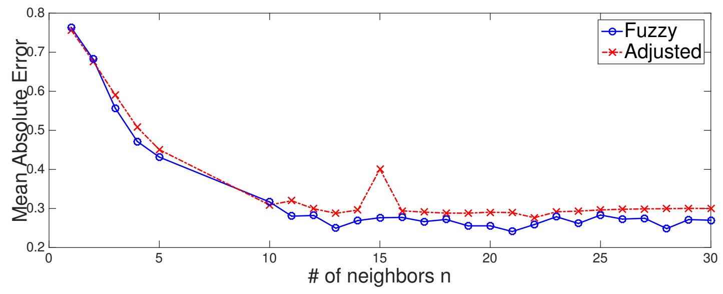

Figure 6: Number of Nearest Neighbors vs Mean Absolute Error for Fuzzy K-means and Adjusted K-means Clustering ($m = 1.45$, $c = 0.2$, $k = 2$)

4.6 Fuzziness Parameter vs MAE

The fuzziness parameter (m) is responsible for assigning cluster membership in Fuzzy K-means and Adjusted K-means algorithms. From Figure 7, it is hard to tell which of the Fuzzy and the Adjusted K-means algorithm perform better with given m. Given the scale on the y-axis, the variations in MAE are noisy and the value of m has little impact. On multiple experimental runs, though, we consistently found that a fuzziness around 1.4 or 1.5 was best, even if marginally so.

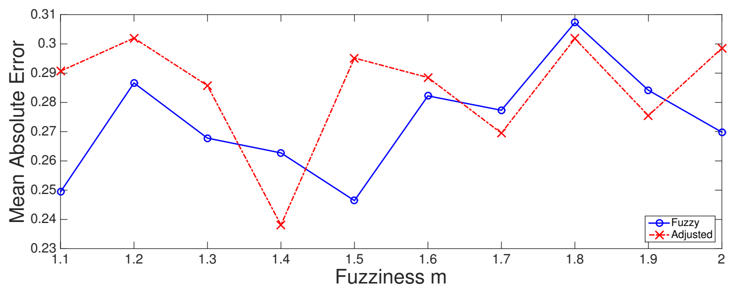

Figure 7: Fuzziness parameter vs Mean Absolute Error for Fuzzy K-means and Adjusted K-means Clustering ($k = 2$, $c = 0.2$, $n = 20$)

4.7 Hybrid System vs Low Rank Matrix Factorization

As seen in Figure 8, the hybrid recommendation system performs as well as the sparse matrix completion based collaborative filtering algorithm. However, the utility of hybrid approach lies in cold start situations or when the size of training data set is low. The complexity of the sparse matrix factorization method grows greatly with the increase in size of training data, a disadvantage compared to hybrid approach.

5 Conclusion

We perform two soft clustering algorithms - Fuzzy and Adjusted K-means- to the item content information to complement the user rating information, which improves the performance of the

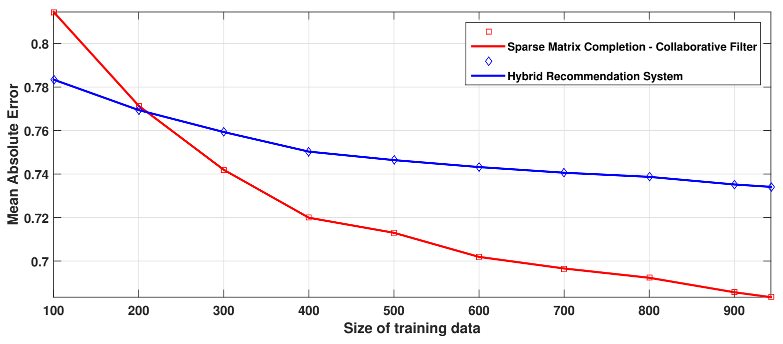

Figure 8: Size of training data vs Mean Absolute Error for sparse matrix completion based collaborative filter and hybrid recommender system

collaborative filter, and also solves the cold start problem (though not presented here). Our work demonstrates that a wise selection of variables like fuzziness parameter, linear combination coefficent for similarity measures, number of clusters, and neighborhood can improve recommendation systems. We also show that using the Adjusted K-means algorithm, we can achieve similar results to the Fuzzy K-means algorithm with lower computational complexity. Finally, we demonstrate that including item content within your formulation can yield much better results than a purely rating-driven collaborative filter.

6 Description of Individual Effort

The project archive was equally distributed among the group members. Firstly, Soumyanil and Ajay implemented fuzzy clustering algorithm, while Gaurav and Kevin implemented adjusted K-means algorithm. The clustering algorithms are used throughout the project. Additionally, Soumyanil and Ajay worked on functions to generate similarity matrices and verified its functionality through a small toy problem. Kevin worked upon utilizing the similarity scores to predict the ratings of a movie based on the ratings for its nearest neighbors. Gaurav implemented the low rank matrix factorization based collaborative filtering algorithm. So overall, two independent recommender systems are proposed and implemented and their performance is compared. It should be noted that the implementation of Equations 1, 5, and 7 took quite longer than expected due to the fact that, in the original paper, it was not explained that when finding the similarity between items k and l, only users that have rated both should be used. It required multiple group meetings spent tinkering to get our results to match Li & Kim's [1]. Because of this, we ended up using the MATLAB built in function for fuzzy clustering to get to the experimental portion faster.

References

[1] Qing Li; Byeong Man Kim, ”Clustering approach for hybrid recommender system,” in Web Intelligence, 2003. WI 2003. Proceedings. IEEE/WIC International Conference on Web Intelligence, pp.33-38, 13-17 Oct. 2003

[2] Miles Trochesset; Anthony Bonner, ”Clustering Labeled Data and Cross-Validation for Classification with Few Positives in Yeast,” in Proceedings of the 4th Workshop on Data Mining in Bioinformatics, 2004.

[3] G.Salton; McGill, ”Introduction to Modern Information Retrieval,” McGraw-Hill, New York, 1983.

[4] D.Goldberg; D.Nichols;B.M.Oki;D.Terry, ”Using Collaborative filtering to weave an information tapestyre,” Communications of the ACM, 35(12):61-70, January 1992.

[5] P.Resnick; N.Iacovou;M.Suchak;P.Bergstorm;J.Riedl, ”Grouplens: An open architecture for collaborative filtering of netnews,” in Proceedings of the ACM Conference on Computer Supported Cooperative Work, 1994.

[6] U.Shardanand; Maes, ”Social information filtering:Algorithms for automating.”

[7] R. Burke, Hybrid recommender systems: survey and experiments : User Modeling and User-Adapted Interaction, 12 (4) (2002), pp. 331370

[8] J.L. Herlocker, J.A. Konstan, J.T. Riedl, An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms, Information Retrieval, 5 (2002), pp. 287310

[9] G. Adomavicius, A. Tuzhilin, 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, 17 (6) (2005), pp. 734749