Yelp Food Recommendation System

Sumedh Sawant
Stanford University
sumedh@stanford.edu

Gina Pai
Stanford University
gpai@stanford.edu

***Abstract*—We apply principles and techniques of recommendation systems to develop a predictive model of customers' restaurant ratings. Using Yelp's dataset, we extract collaborative and content based features to identify customer and restaurant profiles. In particular, we implement singular value decomposition, hybrid cascade of K-nearest neighbor clustering, weighted bi-partite graph projection, and several other learning algorithms. Using Root metrics Mean Squared Error and Mean Absolute Error, we then evaluate and compare the algorithms' performances.**

I. Introduction

A vast database of reviews, ratings, and general information provided by the community about businesses, Yelp provides consumers with a myriad of options and information even when searching for an especially specific service or goods niche. However, although all required information may be present to make an informed choice, it is often still difficult by just looking at the raw data. Reading all the reviews of a single business alone is time consuming and requires more effort than the average user is willing to expend. As a result, we believe users could greatly benefit from a recommendation system.

Recommendation systems have historically been created for various Machine Learning applications in numerous disciplines. One such example is social networking sites such as Facebook that utilize recommendation systems to suggest friendships to users. Music and media applications such as iTunes and Spotify also utilize similar machine learning and recommendation logic to suggest various songs, videos, movies, etc. to users based off their previous choices and taste. Given this general theme, our project focuses on creating a recommendation system for Yelp users in application to potential food choices they could make.

The rise of the popular review site Yelp has led to an influx in data on people's preferences and personalities when it comes to being a modern consumer. Recommendation systems that can identify a user's preferences and identify other similar users' and/or restaurants that match his/her preferences can make this problem easier. Specifically, we aim to build a recommendation system that will enable us to make sophisticated food recommendations for Yelp users by applying learning algorithms to develop a predictive model of customers' restaurant ratings.

We begin by providing a brief explanation of the dataset we used while creating our recommendation system. We follow this with an explanation of the performance metrics we used to evaluate our results. We provide an explanation of our baseline algorithm and its performance, the algorithms we implemented, and the processes we used during our development. Finally, we conclude with a discussion of future work that could be explored.

II. Dataset

Our primary dataset is the Yelp Dataset Challenge data (http://www.yelp.com/dataset_challenge/) that contains actual business, user, and users' review data from the greater Phoenix, AZ metropolitan area. After filtering the data for food-related businesses and reviews, there remain $> 6,000$ businesses, $> 184,000$ reviews, and $> 44,000$ users who gave food reviews. By using and combining various data fields, we identify both similar businesses and users to aggregate the likely sparse ratings per average user.

III. Evaluation Metric

After researching the kinds of evaluation metrics prevalently used to evaluate recommendation systems, we chose to evaluate our system based on the root mean square error (RMSE) and the mean absolute error (MAE) metrics through k-fold cross validation. As the Microsoft Research paper [2] on evaluating recommendation systems argues, both root mean squared error and mean absolute error are popular and highly-accurate metrics historically used to measure the performance of a recommendation system that aims to predict the rating (1 to 5 stars in Yelp) that a particular user would give to an item. Just as the [2] states, during our performance evaluation we assume that our system will generate predicted ratings $\widehat{r}_{uf}$ for a particular user u and food restaurant/location f from our training set S of data (where we know what the actual ratings r_{uf} are). Given this, RMSE error is given by:

$$\text{RMSE} = \sqrt{\frac{1}{|S|} \sum_{(u,f)\in S} (\widehat{r}_{uf} - r_{uf})^2}$$

Also, MAE error is given by:

$$\text{MAE} = \sqrt{\frac{1}{|S|} \sum_{(u,f)\in S} |\widehat{r}_{uf} - r_{uf}|}$$

IV. Baseline

For our baseline, we adopted a similar approach to the AT&T collaborative filtering paper[5]. In particular, we predicted the rating $\widehat{r}_{u,f}$ of a particular user u for a restaurant f as $\widehat{r} = \mu + b_u + b_f$ where μ is the average rating of restaurants by all users, b_u is the difference of user u's average rating from μ, and b_i is the difference of restaurant f's average rating from

μ. In order to make baseline recommendations we calculate $\widehat{r}$ in this manner for all of the restaurants that a user has not reviewed, sort the ratings from greatest to least and output the top 10%.

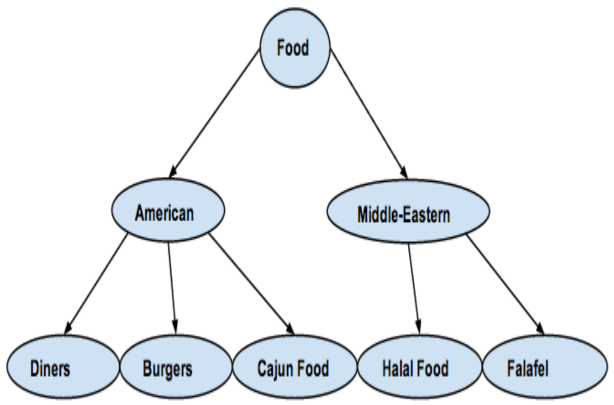
V. Singular Value Decomposition

We applied singular value decomposition[3] as suggested in the recommender systems article [4] to reduce dimensionality of the feature space and make predictions. SVD is a technique used to factor some matrix X into three matrices U, Σ, V^T where Σ is a diagonal matrix with the singular values of X on its diagonal and U and V are orthogonal. Formally, SVD factors $X = U\Sigma V^T$. Following the approach from [4], we created design matrix X where each row represented a user, each column a business, and each entry the rating the user gave the business. For empty entries with no ratings, we populated it with the average rating for the particular business represented by the column. We then ran SVD on this matrix X to get U, Σ, V^T and reformulated an approximation $\widehat{X}$ to X by keeping only the first k singular values in Σ (made it k-by-k and modified U and V^T accordingly) and computed $\widehat{X} = U\Sigma V^T$. Our predicted rating $\widehat{r}_{u,i}$ was the corresponding entry in the matrix $\widehat{X}_{u,i}$.

VI. Hybrid Cascade of KNN

A. K-Nearest Neighbors Clustering on Businesses

The sparseness of ratings per user user makes it very unlikely that any two users had rated the same restaurant. As a result, calculating similarities based solely on shared ratings would likely result in severe over and under-estimations. Thus in order to account for this issue and build a reasonably accurate prediction algorithm, we first clustered the restaurants using knowledge technique. First we formed a category hierarchy by specificity using our understanding of the topics in order to cluster businesses using only its most specific categories. General categories such as "ethnic food" are located at the top of the tree and are parent to those of more specific categories such as "falafel" at the bottom of the tree. Here is a sample of the category hierarchy we generated:



Using our understanding of the category semantics, we next grouped the food categories by theme (Dietary Restrictions, Asian, etc.) and clustered businesses by the group they share the most number of categories with. After forming these initial clusters, we ran k-nearest neighbor clustering algorithm with 10-fold cross validation using a combination similarity function of number of users who rated both businesses and the similarity of those ratings.

B. K-Nearest Neighbors Clustering on Users

Using the results from KNN clustering on restaurants, we changed (and slightly reduced) the feature space of our users by mapping each feature that they had that was specific to a business to a particular business cluster. This enabled us to ensure that a majority of the possible pairs of users had some common particular business cluster that they had both rated, ultimately allowing us to more accurately calculate similarity between users. Thus once we had created this new input feature space of users, we ran k-nearest neighbors with 10-fold cross validation on the users to group them into clusters of similar users using a similarity function based on the similarity of the ratings.

Once using hybrid cascading to create the similar user clusters, to predict a rating $\widehat{r}_{u,f}$ of a particular business f for a user u, average together all of the cluster's ratings for the business cluster. First we mapped the business to its specific cluster C_f. Next, we checked which users in the cluster C_u of user u had rated food from businesses in cluster C_f and averaged together all of the ratings. This served as our predicted rating for user u for the business f.

$$\widehat{r}_{u,f} = \frac{1}{|R_{C_u,C_f}|} \sum_{k \in R_{C_u,C_f}} k$$

In the expression above, R_{C_u,C_f} is the set of all ratings by users in C_u of businesses in C_f. The results of implementing this prediction system are given below in the results section.

To generate recommendations for a user using this final model, predict ratings in a similar fashion on all businesses the user had not reviewed yet and output the ones that yield the highest rating%.

VII. Weighted BiPartite Graph Projection

By utilizing our representation of the Yelp dataset as a weighted bipartite graph where edges from user to business are weighted by rating, we posed the recommendation problem as graph projection by using the same novel network-based-inference collaborative filtering algorithm that was proposed by [7] and originally created by [6]. More specifically we followed a network based resource allocation process to produce similarity measures (which can be thought of as weights in a bipartite graph projection) between every pair of users and every pair of food businesses, which are then used to produce predicted ratings and recommendations.

We used a neighborhood-based collaborative filtering approach with our novel similarity function based off network inference in order to make predictions. We chose to only explain the user-based approach to neighborhood collaborative filtering here since the food business version would be mathematically equivalent. Thus, our user-based collaborative recommendation approach aims to calculate some similarity metric between all pairs of users and then predict a particular user u's rating for a food business b by collecting and processing the ratings for b of u's neighborhood (all the other users with high similarity as compared to u). Formally, we first calculated and stored the value of some similarity metric between any two pair of users, and then computed the prediction of the rating of user u towards a food business b by computing a weighted average of u's neighbors ratings of b (the weights are the similarity between u and each neighbor).This is expressed as ([8]):

$$\widehat{r}_{u,b} = \overline{r}_u + \kappa \sum_{j=1}^{n} sim(u,j)(r_{j,b} - \overline{r}_j)$$

$$\overline{r}_u = \tfrac{1}{B_u} \sum_{j \in B_u} r_{u,j}$$

Here $\overline{r}_u$ is the average rating given by user u (B_u above is the set of food businesses on which user u voted), $sim(u,j)$ is the similarity measure between user u and user j, κ is a normalizing factor so that the absolute values of the similarity metrics sum to 1, and $r_{u,b}$ is the actual rating given by user u to business b (note that this is different from $\widehat{r}_{u,b}$ which is the predicted rating).

We defined a new $sim(x,y)$ function called recommendation power. As [6] states, the weight w_{ij} in a weighted bipartite graph projection can be thought of as how important node j is to node i (thus $w_{ij} = w_{ji}$ does not always hold). We can intuitively think of this as each user giving his or her neighbors some amount of "recommendation power" that they in turn can use to recommend food businesses. The more "recommendation power" that a user has, the more powerful or influential his or her recommendation is to the original user who gave them that power.

In order to formalize this notion of distributing the recommendation power (or resource), as done in [7] we can consider a 2-step random walk on the Yelp bipartite graph where we walk 2 steps starting from some user node, going to a business node, and coming back to a different user node. Each step that we take has some transition probability of occurring. Intuitively, for the first step in this process we can think of a user as being more likely to give a particular food business some of his or her resource if the user rated that business highly. Thus we can think of the probability of the transition from user u to food business b as being $\frac{r_{u,b}}{R_u}$ where $r_{u,b}$ is the rating that user u gave food business b and R_u is the sum of ratings that user u ever gave. Similarly, we can think of a food business b as being more likely to give a user v some of the resource it received if user v rated food business b highly. Thus we can think of the probability of the transition from business b to user v as being $\frac{r_{v,b}}{R_b}$ where $r_{v,b}$ is the rating that user v gave business b and R_b is the sum of ratings that business b ever received. Thus the value $\frac{r_{u,b}}{R_u}\frac{r_{v,b}}{R_b}$ is the probability of a transition from user u to a particular food business b and back to user v (and thus the amount of recommendation power or resource that user v receives from user u through business b in the network). Since user u distributes resource to all of the businesses and the expression $\frac{r_{u,b}}{R_u}\frac{r_{v,b}}{R_b}$ represents the amount of resource user v receives from user u from one particular food business, we can sum over all businesses to get the total amount of resource that user v receives from user u. Thus we have that

$$rp(u,v) = \sum_{b \in B} \frac{r_{u,b}}{R_u}\frac{r_{v,b}}{R_b}$$

Thus, substituting into our formula from above for collaborative filtering, a prediction $\widehat{r}_{u,b}$ of a user u's rating for a food business b can be made as $\widehat{r}_{u,b} = \overline{r}_u + \sum_{v \in U} rp(u,v)(r_{v,b} - \overline{r}_v)$ where $\overline{r}_u = \frac{1}{B_u} \sum_{b \in B_u} r_{u,b}$. Also notice that $\kappa = 1$ in the formula since $\sum_{v \in U} rp(u,j) = 1$ due to the probabilistic properties of $rp(u,v)$. We used this approach to make recommendations for our system.

VIII. Clustered Weighted BiPartite Graph Projection

While the "Weighted BiPartite Graph Projection" algorithm above improved our prediction accuracy, our dataset suffered from a sparseness problem. In essence, the bipartite graph representation did not have many scenarios where two users had rated the same food business. The "Weighted BiPartite Graph Projection" algorithm was not robust to this and ultimately lost some accuracy since the similarity measures it calculated depended on the dataset having many scenarios where two users had rated the same business. In order to improve upon this and the general accuracy of the algorithm we created an experimental learning algorithm algorithm that is an extension called "Clustered Weighted BiPartite Graph Projection":

1) Use the k-means clustering algorithm using the recommendation power metric as similarity difference (instead of the norm) to partition the users into a set C_u of k_1 user clusters and partition the businesses into a set C_b of k_2 business clusters.

2) Construct a compressed version G' of the original bipartite graph G by creating a set of nodes representing each C_u user cluster and a set of nodes representing each C_b cluster. An edge from a node representing cluster C_u to a node representing cluster C_b exists if any user from C_u ever rated a business in C_b.

3) Run the same weighted bipartite project algorithm from [2] on the compressed graph G' and predict the rating that a user u would give a business b by simply considering the rating that the node C_u would give the node C_b in the compressed graph.

In this experiment, we found that with the right amount of clustering on the dataset, the intolerance to sparseness by the "Weighted BiPartite Graph Projection" algorithm above could be combated by essentially transforming our sparse dataset into our own non-sparse definition through some similarity mapping and grouping. Once we had a non-sparse dataset, the "Weighted BiPartite Graph Projection" algorithm above worked just as it should and gave very accurate results.

We experimented with the values k_1, k_2, which were the number of user clusters and business clusters respectively. When we had a really few amount of clusters, we were not capturing enough of the differences between each of the users (or businesses) and thus underfit the data. As we created more and more clusters by increasing k_1, k_2, we captured more and more of the data's specifics and thus were able to make better recommendations. Eventually we reached some critical point where we had created the optimal number of clusters to drive down the error. If we created any more clusters than the optimal amount, we approached the scenario where the clusters were so small that we started to capture the really minor differences between the users and treated them as huge differences in making recommendations. Thus by making more than the optimal amount of clusters, we overfitted the dataset and started to increase the error metric. The user cluster amount k_1 had a higher optimal value than the business cluster amount k_2 because there were more users than businesses in the dataset and there was more variance in the type of user. Thus we needed more clusters to optimally capture the different types of users than we did to optimally capture the different types of businesses.

IX. Multi-Step Random Walk Weighted BiPartite Graph Projection

The "Weighted BiPartite Graph Projection" algorithm that we use in this paper was derived from using a 2-step random walk on the bipartite graph to allocate the recommendation power of a user u to another user v. However instead of just using a 2-step random walk, in this experiment we used a summation of the results of multiple random walks (2-step, 4-step, ..., k-step where k is an even number greater than 2) where the starting point of each walk is always a particular node u and the ending point of each walk is always a particular node v (each walk contributes a little bit of recommendation power to v from u). Thus mathematically we had:

$$rp(u,v) = \sum_{b\in B} \frac{r_{u,b}}{R_u}\frac{r_{v,b}}{R_b} + \alpha\Big(\sum_{b_1,b_2\in B} \frac{r_{u,b_1}}{R_u}\frac{r_{k_1,b_1}}{R_{b_1}}\frac{r_{k_1,b_2}}{R_{k_1}}\frac{r_{v,b_2}}{R_{b_2}}\Big) +$$
$$\alpha^2\Big(\sum_{b_1,b_2\in B} \frac{r_{u,b_1}}{R_u}\frac{r_{k_1,b_1}}{R_{b_1}}\frac{r_{k_1,b_2}}{R_{k_1}}\frac{r_{k_2,b_2}}{R_{b_2}}\frac{r_{k_2,b_3}}{R_{k_2}}\frac{r_{v,b_3}}{R_{b_3}}\Big) +$$
$$\dots + \alpha^n\Big(\sum_{b_1,\dots,b_n\in B} \frac{r_{u,b_1}}{R_u}\frac{r_{k_1,b_1}}{R_{b_1}}\cdots\frac{r_{k_{n-1},b_n}}{R_{k_{n-1}}}\frac{r_{v,b_n}}{R_{b_n}}\Big)$$

Note that in the equation above $n = k/2 - 1$. Also $k_1, \dots, k_n$ represent arbitrary users in our network. Also, α is a decaying factor that is applied to the result of each random walk to ensure that $\sum\limits_{v\in U} rp(u,j) = 1$. Using this new definition of $rp(u,v)$ as our new similarity function, a prediction $\widehat{r}_{u,b}$ of a user u's rating for a food business b can be made as $\widehat{r}_{u,b} = \overline{r}_u + \sum\limits_{v\in U} rp(u,v)(r_{v,b} - \overline{r}_v)$ where $\overline{r}_u = \frac{1}{B_u}\sum\limits_{b\in B_u} r_{u,b}$.

Because the multi-step random walk approach involved the summation of the results of multiple random walks, it was extremely inefficient to compute. Looking at the mathematical expression for the recommendation power between two users (or equivalently food businesses), we can see that a k-step random walk involves a summation over $k/2 - 1$ different combinations of user (or food business) variables, making its time complexity $O(n^{k/2-1})$ since we need to iterate over the entire set of users (or businesses) for every variable in the summation bounds. This can be extremely expensive on large datasets. In essence, we wanted to avoid having to calculate random walks for large values of k for our Yelp dataset. Thus we simply looked at the value of k on the graph above where the error started to converge and used that value of k as our "most accurate" k value. We ran several trials of these multi-step random walks (each with a different value of k) to make recommendations and measured the RMSE and MAE error. We found that the error values dropped and tended to converge after about $k = 10$ iterations, so when we actually ran this algorithm and displayed its results below, we used a value of $k = 10$.

X. Cascaded Clustered Multi-Step Weighted BiPartite Graph Projection

We also implemented a hybrid version of the previous two mentioned algorithms by trying to take advantage of both types of strategies for making the weighted bipartite graph projection algorithm better. Thus we proceed as follows:

1) Use the k-means clustering algorithm using the recommendation power metric as similarity difference (instead of the norm) to partition the users into a set C_u of k_1 user clusters and partition the businesses into a set C_b of k_2 business clusters.

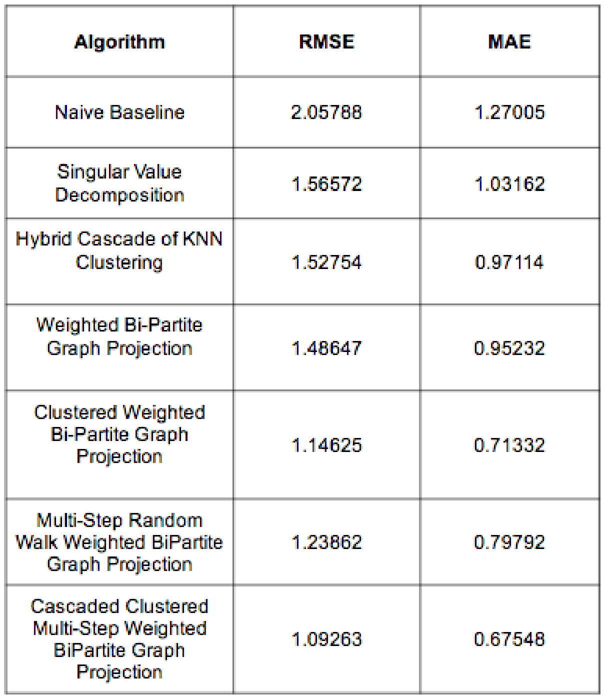
2) Construct a compressed version G' of the original bipartite graph G by creating a set of nodes representing each C_u user cluster and a set of nodes representing each C_b cluster. An edge from a node representing cluster C_u to a node representing cluster C_b exists if any user from C_u ever rated a business in C_b.

3) Run the multi-step random walk based algorithm from above on the compressed graph to make predictions.

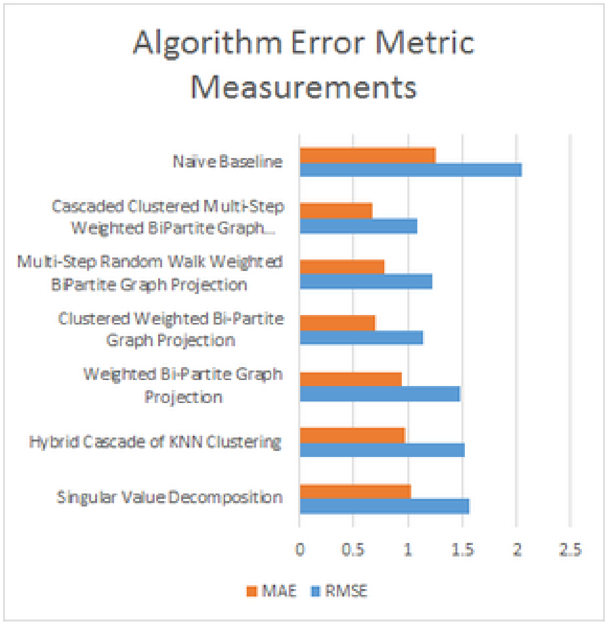
When implementing this algorithm, after we had generated the compressed graph, we ran several trials of the multi-step random walks (each with a different value of k) to make recommendations and measured the RMSE and MAE error. We found that the error values dropped and tended to converge after about $k = 6$ iterations, so when we actually ran this algorithm and displayed its results below, we used a value of $k = 6$. This algorithm was more accurate than either of the individual algorithms it was based on since we took advantage of the fact that the compressed graph accounts for the sparseness of the graph, and consequently that the similarity measure between two entities on the compressed graph means more than it did with the original graph. Because of this, as we continued to improve the similarity measure to be more and more accurate on the compressed graph by using the combined results of multi-step random walks, we got recommendations that were even more accurate than before, effectively combining the sparseness-fighting effects of clustering and the similarity-precision of multi-step random walks in order to collaboratively create a very accurate prediction model.

XI. Results

Here are the results from running each implemented algorithm on our dataset. As mentioned above, we used k-fold cross validation to evaluate the error of the recommendation system for each algorithm in terms of RMSE and MAE.



Algorithm	RMSE	MAE
Naive Baseline	2.05788	1.27005
Singular Value Decomposition	1.56572	1.03162
Hybrid Cascade of KNN Clustering	1.52754	0.97114
Weighted Bi-Partite Graph Projection	1.48647	0.95232
Clustered Weighted Bi-Partite Graph Projection	1.14625	0.71332
Multi-Step Random Walk Weighted BiPartite Graph Projection	1.23862	0.79792
Cascaded Clustered Multi-Step Weighted BiPartite Graph Projection	1.09263	0.67548



As we can see from the results, the "Cascaded Clustered Multi-Step Weighted BiPartite Graph Projection" algorithm performs the best of all the algorithms, both combating the sparseness of the dataset and utilizing network based inference to make accurate predictions.

XII. Future Work

In the future, we would augment the current analysis to include review text and user rating evaluations (whether other users thought a particular user's review was funny, useful, or helpful) as features in the prediction model. We would also explore further hybrid approaches and evaluate their performances.

References

[1] Burke, R., *Hybrid Recommender Systems: Survey and Experiments*, <http://josquin.cs.depaul.edu/∼ rburke/pubs/burke-umuai02.pdf>.

[2] Gunawardana A., Shani G, *Evaluating Recommendation Systems*, <http://research.microsoft.com/pubs/115396/evaluationmetrics.tr.pdf>.

[3] Wikipedia, *Singular Value Decomposition*, <http://en.wikipedia.org/wiki/Singular_value_decomposition>.

[4] Recommender Systems, *Dimensionality Reduction and the Singular Value Decomposition*, <http://www.cs.carleton.edu/cs_comps/0607/recommend/recommender/svd.html>.

[5] Koren, Y., *Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model*, <http://public.research.att.com/∼volinsky/netflix/kdd08koren.pdf>.

[6] Zhou, T., et al., *Bipartite network projection and personal recommendation*, Physical Review E, 2007. 76(046115)

[7] Shang, M., Fu, Y., Chen, D., *Personal Recommendation Using Weighted BiPartite Graph Projection*, Apperceiving Computing and Intelligence Analysis, 2008. ICACIA 2008. International Conference on , vol., no., pp.198,202, 13-15 Dec. 2008

[8] Breese, J.S., Heckerman, D., Kadie, C., *Empirical Analysis of Predictive Algorithms for Collaborative Filtering*, <http://research.microsoft.com/pubs/69656/tr-98-12.pdf>