A Random Walk Method for Alleviating the Sparsity Problem in Collaborative Filtering

Hilmi Yıldırım and Mukkai S. Krishnamoorthy
Rensselaer Polytechnic Institute
Computer Science Department
Troy, New York
{yildih2,moorthy}@cs.rpi.edu

ABSTRACT

Collaborative Filtering is one of the most widely used approaches in recommendation systems which predicts user preferences by learning past user-item relationships. In recent years, item-oriented collaborative filtering methods came into prominence as they are more scalable compared to user-oriented methods. Item-oriented methods discover item-item relationships from the training data and use these relations to compute predictions. In this paper, we propose a novel item-oriented algorithm, Random Walk Recommender, that first infers transition probabilities between items based on their similarities and models finite length random walks on the item space to compute predictions. This method is especially useful when training data is less than plentiful, namely when typical similarity measures fail to capture actual relationships between items. Aside from the proposed prediction algorithm, the final transition probability matrix computed in one of the intermediate steps can be used as an item similarity matrix in typical item-oriented approaches. Thus, this paper suggests a method to enhance similarity matrices under sparse data as well. Experiments on MovieLens data show that Random Walk Recommender algorithm outperforms two other item-oriented methods in different sparsity levels while having the best performance difference in sparse datasets.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval —*Information Filtering*

General Terms

Algorithms, Experimentation, Performance

Keywords

collaborative filtering, sparsity, random walk

1. INTRODUCTION

Recommendation systems became an important research area with the enormous expansion of Internet services such as e-commerce sites. With the increase in diversity of products and services, consumers have gained the opportunity to find ideal items along with the drawback of selecting among abundance of uninteresting products. As an example, Amazon.com provides millions of books and other products in which a typical user might have no interest. Depending on the past transactions of a user, recommending products of interest to the user may both be useful to the system and to the user. Within this concept, many online businesses such as Amazon.com and Netflix (an online movie renter) have used recommendation systems to provide personalized suggestions. Such systems are reported to boost online sales by converting shoppers to buyers and building customer loyalty[12]. In October 2006, Netflix announced its one million dollar prize on recommendation systems which sparked researchers' interests.

In a broad view, recommendation systems keep track of past purchases of customers as well as other information of product and customer profiles to make personalized recommendations for each customer. In the most common formulation, the recommendation problem is reduced to the problem of estimating ratings for the products that have not been seen by a user. However, based on the nature of the application, recommendation problem can be thought as identifying only the highest ranked products for each user instead of predicting the rating of each product for each user.

Collaborative Filtering(CF) [1] is one category of recommendation systems which solely relies on the past behavior (ratings, purchase history, time spent) of the users. As opposed to content based approaches, CF does not use explicit profiles of items (i.e., keywords that describe the items, such as artists, genres, etc., for movies) and users. CF has many advantages over content-based methods such as its genericity and its potential to explore implicit associations. For example in some domains such as video recommendation, it might not be easy to build item profiles whereas CF methods can be applied to all domains. Thus there is a wealth of empirical [6, 7, 8, 10, 17, 19, 20, 21] and theoretical [3, 9, 14, 15, 16] research on CF resulting in many successful commercial products such as Amazon.com and Netflix. Section 2 gives a detailed survey on that literature.

[1]We use the term CF as an abbreviation of Collaboration Filtering throughout the paper.

CF methods can be classified into two common classes. The first approach, known as memory-based, is based on the assumption that each user belongs to a larger group of similarly behaving users. Therefore if similar users can be identified for a particular user, a combination of the preferences of that group gives reasonable recommendations for the user in question. Indeed this method is referred to as user-oriented memory-based approach; an analogous method which builds item similarity groups using co-purchase history is known as item-oriented. The second approach, referred to as model-based, assumes a model which generates the ratings and learns that model with training data. Clustering, Bayesian network models and other machine learning techniques can be listed under model-based approaches[2, 7].

The focus of this paper is on a model-based approach that especially works well in sparse data. The method described here can be viewed as an extension to the *item-based top-N recommendation* algorithm [8] in the sense that the way it builds and uses item similarities to infer recommendations. *Item-based top-N recommendations* algorithms have been used in various forms since the early days of CF recommender systems[21]. Although most of the successful CF methods rely on building user-user relationship, unfortunately these methods are not scalable since in a typical commercial system number of users can be much larger than the number of items. In [20],the authors show empirically that item based methods can perform as good as user based methods. On the other hand, our method has resemblance to *Huang et al.*'s link analysis approach [13] in the manner that they deal with sparsity problem.

We propose a model in which users make random walks in the item space according to the similarity of items. An initial normalized item-similarity matrix, which can be obtained in various ways [2, 20], is interpreted as a transition probability matrix. Therefore, a user enjoying an item most likely jumps to a similar item in the next step of his walk. The aggregate probability distribution of a user over the items throughout this random walk constitutes the final rating predictions for the items. This method has three key steps, which are (i) building the initial transition probability matrix, (ii) obtaining the aggregate probability distributions and (iii) interpreting these probabilities to predict ratings. This method outperforms *Deshpande et al.*'s item-based recommendation algorithm where the training data is less than plentiful. Furthermore, final transition probability matrix obtained in this method can be used as item-similarity matrix in other item-oriented approaches when the data is sparse. The main contribution of this paper, indeed, is the method to obtain an item-similarity matrix which also takes into account transitive item associations rather than just co-purchase history.

The remainder of the paper is organized as follows: Section 2 provides the definitions and the historical review of the literature on collaborative filtering. Section 3 describes the model and the algorithm used to predict ratings for each item. Section 4 provides an experimental evaluation of the algorithm in different sparsity levels and shows the implications of various parameters on the performance. This section also gives a brief comparison against another item-oriented approach and a method which was proposed to alleviate sparsity problem. Finally, section 5 concludes with discussions and possible improvements.

2. HISTORICAL REVIEW

2.1 Definitions and Notations

There exists a set of U users and a set of I items where $|U| = n$ and $|I| = m$. Each user has a numeric value for each product. Some of the papers assume only binary values (e.g., good and bad) as ratings for the sake of simplicity. These values constitute an n x m matrix, $R = \{r_{ui}\}_{1\leq u\leq n, 1\leq i\leq m}$ whose rows correspond to user preference vectors. Additionally, $\vec{R}_{u\_}$ is the rating vector of user u, whereas $\vec{R}_{\_i}$ is the rating vector for item i. Throughout the paper, we use letters of u, v for indexing users and i, j for items for the sake of consistency.

Prediction Problem : Given a rating matrix R with some unknown entries, predict each missing entry such that the difference between the predicted matrix and the actual matrix is minimized.

2.2 Collaborative Filtering

CF is the method of making predictions on the ratings of a particular user by collecting taste information from many users. The term was first used by *Goldberg et al.* in the Tapestry system which relies on each user to identify similar-minded users manually. In [10] *Breese et al.*, authors categorized CF into two general classes of *memory-based* and *model-based* methods[7].

2.2.1 Memory Based Approaches

Memory-based algorithms[7, 19, 21] are heuristics to predict a missing rating by aggregating the ratings of k users who are most similar to a particular user. This can also be seen as identifying the k-Nearest Neighbors and combining their information to predict a rating. Formally, a missing rating $r_{u,i}$ can be formulated as

$$r_{ui} = \operatorname*{aggr}_{v\in U'} r_{vi} \tag{1}$$

where U' is the set of users who are most similar to user u who have rated i, and *aggr* is the method of combining these values. In the simplest case, the aggregation is the average of ratings of similar users on a particular item [2].

Analogously the rating $r_{u,i}$ can be predicted as an aggregation of items similar to i and which are rated by u. The first approach is known as user-oriented whereas the latter is cited as item-oriented. In [20],the author shows that item-oriented approaches perform as good as the other approaches while being more scalable and efficient.

To identify the neighborhoods of users, these methods compute similarity between each pair of users. Common choices are the cosine similarity and the Pearson correlation approaches. In the cosine based approach[7, 20], the similarity between u and v, s_{uv}, is measured by computing the cosine angles between the rating vectors of users u and v. Only the items rated by both users are taken into account while computing the angles. Formally if we assume S_{uv} is the set of items which are rated by both of the users, the similarity s_{uv} is:

$$s_{uv} = cos(\vec{u}, \vec{v}) = \frac{\sum_{i\in S_{uv}} r_{ui}r_{vi}}{\sqrt{\sum_{i\in S_{uv}} r_{ui}^2}\sqrt{\sum_{i\in S_{uv}} r_{vi}^2}} \tag{2}$$

Correlation based approach uses Pearson correlation coefficient to measure the similarities between users [19, 21]. The advantage of this method is that it also takes into account that different users might have different rating schemes.

$$s_{uv} = \frac{\sum\limits_{i \in S_{uv}} \left(r_{ui} - \overline{r_u}\right)\left(r_{vi} - \overline{r_v}\right)}{\sqrt{\sum\limits_{i \in S_{uv}} \left(r_{ui} - \overline{r_u}\right)^2} \sqrt{\sum\limits_{i \in S_{uv}} \left(r_{vi} - \overline{r_v}\right)^2}} \qquad (3)$$

where $\overline{r_u}$ is the average of nonzero ratings of the user u.

Second step of memory-based methods is computing the rating, r_{ui}, using the ratings of most similar neighbors of user u. The key point is the aggregation function shown in equation (1). This function usually takes the weighted average of neighbors, which implies

$$r_{ui} = \frac{\sum\limits_{v \in N(u,i)} w_{uv} r_{vi}}{\sum\limits_{v \in N(u,i)} w_{uv}} \qquad (4)$$

where $N(u,i)$ is the set of closest neighbors of u who have rated i, and w_{uv} is the weight of similarities in user-oriented approach. Analogously, the item-oriented version of rating interpolation can be defined as

$$r_{ui} = \frac{\sum\limits_{j \in N(i,u)} w_{ij} r_{uj}}{\sum\limits_{j \in N(i,u)} w_{ij}} \qquad (5)$$

where $N(i,u)$ is the set of closest neighbors of item i which are rated by u, and w_{ij} is the weight of similarities.

Resnick et al. and *Shardanand* used similarity values between users directly as interpolation weights in their user-oriented approaches (i.e., they assumed $w_{uv} = s_{uv}$) [19, 21]. Later *Sarwar et al.* tailored this intuitive way to their item-oriented approach [20]. Recently, *Bell et al.*, who are the leaders in the Netflix challenge as of the writing of this paper, argued that the heuristic nature of determining interpolation weights needs to be addressed. Different algorithms use different similarity measures and there is no fundamental justification for the chosen similarities. The common approach, using similarity values directly as weights, might ignore or magnify some hidden item dependencies. Thus, *Bell et al.* proposes a formal way to adjust these interpolation weights. Their method identifies similarity groups for each item and learns the actual weights by using the known ratings of users who rated all the items in that neighborhood. The problem turns into a quadratic optimization problem and this approach improves the accuracy of the k-NN significantly without increasing the computational complexity of the overall method [6].

2.2.2 Model Based Approaches

Model-based algorithms use the collection of ratings to learn a model, which is then used to make predictions. There are various approaches under this heading. Clustering models[7, 16, 19, 22], probabilistic approaches[3, 7, 16, 14] and dimensionality reduction and matrix reconstruction techniques [4, 6, 9, 11, 5] can be listed as main model-based approaches.

Breese et al. formulated the prediction problem in a probabilistic manner as the estimated probability of a rating given the known ratings.

$$r_{ui} = E\left(r_{ui}\right) = \sum_{k=0}^{d} k \times Pr\left(r_{ui} = k | r_{uj}, j \in S_u\right) \qquad (6)$$

To estimate this probability, they proposed cluster models and Bayesian networks. The first method clusters the like-minded users and learns the number of classes and the parameters from the data. One limitation of this approach is that each user belongs to a single cluster, whereas some applications may benefit from the ability to cluster users into several categories [2]. Kumar et al.[16] studied the model where there are clusters of products and each user has a probability distribution over clusters. A user first chooses a cluster by his distribution and then uniformly randomly selects an item from that cluster. This method reduces the preference matrix to the size of $n \times k$ where k is the number of clusters. Kleinberg and Sandler [15] generalized this model to the case where the choice within a cluster is arbitrary, and considered a mixture model in which each cluster is a probability distribution over all items.

Azar et al.[4] identified matrix reconstruction as central to recommendation systems. The aim is to reconstruct the whole matrix which can be approximated by a low-rank matrix reconstruction using SVD.[2] Drineas et al. [9] improves this result by reducing the number of samples required; but addresses only the recommendation problem instead of prediction problem. Their algorithm asks a small number(k) of users about all items, and the remaining users for their preference on a small number of items. The requirement that a set of users should vote on all items is not practical in actual recommendation systems. Similarly, Goldberg et al.'s[11] SVD-based Eigentaste algorithm uses a gauge set of items each of which should be voted by each user. Drineas' algorithm works only when a few severe assumptions are met. This method assumes that there are k dominant types, and that their canonical vectors are nearly orthogonal, which means that their preferences don't intersect. The second is gap assumption which is any non-dominant type is far less popular than dominant types. Awerbuch et al.[3] avoids these severe conditions by proposing a combinatorial algorithm without using SVD. Although their method also requires committee members who should vote for each item, it removes the severe conditions of type separability and gap assumption.

3. OUR CONTRIBUTIONS

In this section, we study a class of model-based item-oriented prediction algorithms. The primary motivation behind these algorithms is the success of link analysis methods like Pagerank[18] on ranking items. Pagerank models the random walk of users on the entire web graph (extracted by treating web pages as nodes and the links between pages as edges). It finally assigns each page p a rank of its importance. Specifically the Pagerank of a page p is the probability of visiting p in a random walk of web graph where each random step is either (i, with small probability) jumping to an arbitrary page with uniform probability or (ii, with high probability) walking through one of the outgoing links of the current page. In our application, we develop an item graph where the nodes are items and the edges between nodes represent similarities of items. Furthermore users are assumed to make a finite-length walk as a difference to Pagerank method. Thus ranking becomes dependent on the initial

[2]SVD is an abbreviation for singular value decomposition and it has several applications including matrix approximation.

distribution of the user. We denote the rank of an item i for user u as $rank(u,i)$. These algorithms are similar in spirit to *Deshpande et al.*'s *Item-Based top-N Recommendation Algorithm*[8] in the sense that both produce a ranking of items and similar to *Huang et al.*'s[13] link analysis approach which is proposed to overcome sparseness.

At a high level, our algorithm consists of three main components. The first component is building the item graph which captures the similarity of items between each other. The second component computes the rank values of items for each user by simulating a random walk. Finally the last component interprets and scales the rank scores as ratings for each user-item pair. The details of the algorithm explained in the rest of this section.

3.1 The Model

The model used by our algorithm is a Markov chain model where the probability of being in a state solely depends on the previous step, $\Pr(X_{u,k+1} = i \mid X_{u,k})$. $X_{u,k}$ is the random variable for the user u being at an item i in step k of his random walk. Furthermore this model incorporates a stopping probability to bring an end to the random walk. In each step of the walk user decides to continue his walk with probability α. Therefore the length of the random walks are geometrically distributed with the parameter $1-\alpha$.

We build an item graph P of size m, where the weight of the edge between nodes i and j is equal to the probability of passing from item i to item j of a user.

$$P_{ij} = \Pr(X_{u,k+1} = j \mid X_{u,k} = i) \tag{7}$$

The normalized row vectors of the rating matrix, $\vec{R}^*_{u\_}$, represent the initial distribution of the user on items, namely $\Pr(X_{u,0} = i) = R^*_{ui}$. Thus the probability of user u being in item j in step k is:

$$\begin{aligned}\Pr(X_{u,k} = j) &= \alpha \sum_{i=1}^{m} \Pr(X_{u,k-1} = i)\, P_{ij} \\ &= \alpha^k \sum_{i=1}^{m} R^*_{ui} P^k_{ij} \\ &= \alpha^k \vec{R}^*_{u\_} \cdot \vec{P}^k_{\_j}\end{aligned} \tag{8}$$

We now define the overall probability of user u being at item i which is directly proportional to the sum of probabilities over all steps of the random walk.

$$\begin{aligned}\Pr(X_u = j) &= \frac{\sum\limits_{k=1}^{\infty} \Pr(X_{u,k} = j)}{\sum\limits_{k=1}^{\infty}\sum\limits_{i=1}^{m} \Pr(X_{u,k} = i)} \\ &= c \sum_{k=1}^{\infty} \alpha^k \vec{R}^*_{u\_} \cdot \vec{P}^k_{\_j}\end{aligned} \tag{9}$$

where c is a constant. Since we won't be using actual probabilities, we'll assume $rank(u,j)$ equal to $\sum_{k=1}^{\infty} \alpha^k \vec{R}^*_{u\_} \cdot \vec{P}^k_{\_j}$. Thus the rank matrix, $\tilde{R} = \sum_{k=1}^{\infty} \alpha^k R P^k$ contains the rank scores for all user-item pairs.

$$\begin{aligned}\tilde{R} &= \sum_{k=1}^{\infty} \alpha^k R P^k \\ &= R\alpha P (I - \alpha P)^{-1}\end{aligned} \tag{10}$$

In this setting, rank scores $\tilde{R}$ can be computed in a very similar way to *Item-Based Top-N Recommendation* method in which rank matrix is the direct multiplication of rating matrix and the similarity matrix, namely $R \times S$. The fact that motivates us to bring the idea of random walk approach is that the similarity matrices are usually too sparse to capture actual dependencies between items. For example, an item i that hasn't been rated by any user who has rated item j implies that the similarity score of 0 in most of the item-item similarity measures. However these items would be found as closely to each other, if there is another item t which is similar to both items. Random Walk Recommender captures these transitive associations in various levels proportional to the length of the random walk. Therefore we parametrize the length of the walk according to the sparsity level of the rating matrix and fortunately experimental results confirm our choice that α increases as the initial rating matrix becomes sparser.

Algorithm 1 RandomWalkRecommender(R, α)

```
1: $S \leftarrow$ComputeSimilarityMatrix($R$)
2: for $i \leftarrow 1$ to $m$ do
3:     $sum \leftarrow \sum_{j=1}^{m} S_{ij}$
4:     for $j \leftarrow 1$ to $m$ do
5:         $P_{ij} \leftarrow \beta S_{ij}/sum + (1-\beta)/m$
6:     end for
7: end for
8: $\tilde{P} \leftarrow \alpha P (I - \alpha P)^{-1}$
9: $\tilde{R} \leftarrow R\tilde{P}$
10: Predictions $\leftarrow$ Scale($\tilde{R}$)
11: return Predictions
```

The input to *Random Walk Recommender Algorithm* is the rating matrix, R, and the damping factor α. The output is the *Predictions* which has the same dimensions with R. The actual algorithm is quite simple. Lines 1 through 7 build the transition probability matrix of the Markov chain model. The first line computes the similarities between items according to their co-rate history. There are various methods for this computation in the literature of collaborative filtering and these are explained in the next section. The following lines build the P matrix by considering a user either (i, with probability β) walks through a direct neighbor with probability proportional to the similarity of items or (ii, with probability $1-\beta$) jumps to an arbitrary item with uniform probability. Line 8 and 9 do the actual computation for identifying the ranks of items for each user. Finally the last line scales the ratings such that each row's greatest item has the rating 5, namely rows are scaled independently from each other.

3.2 Computing Transition Probabilities

Transition probabilities are computed in the initialization step of the algorithm and have a critical role in the effectiveness of the approach. As explained in the description of algorithm 3.1, transition probabilities basically depend on the similarity measures between items which are usually identified by analyzing co-rate history. We discussed two of the common approaches to measuring item similarities in section 2.2.1. The following subsections also provide the details of the approaches we used in our implementation.

The similarity measure between items i and j is usually high if there are lots of users who have rated both of the items. The basic idea in similarity computation between i and j is to first identify the users who have rated both of the items and then to apply a similarity computation technique to determine S_{ij}. Common approaches produce symmetric similarity matrices; but we destroy symmetry while computing transition probabilities by normalizing the sum of each row to 1. If an item i is similar to only one item j whereas j has many more similar items, then a random walk from i to j happens with higher probability than the walk in the reverse direction. Breaking the symmetry makes the model to produce higher ratings for frequent items [8]. We also incorporate a small probability to jump to an arbitrary item while computing transition probabilities.

3.2.1 *Cosine Based Similarity*

As introduced in section 2.2.1 and in the equation (2), cosine-based similarity is the cosine of the angles between the projected rating vectors of the items on the users who have rated both of them. The item oriented version of (2) is as follows:

$$S_{ij} = cos(\vec{i}, \vec{j}) = \frac{\sum\limits_{u \in S'_{ij}} r_{ui} r_{uj}}{\sqrt{\sum\limits_{u \in S'_{ij}} r_{ui}^2} \sqrt{\sum\limits_{u \in S'_{ij}} r_{uj}^2}} \quad (11)$$

where S'_{ij} is the set of users who rated both of the items i and j.

The weakness of the cosine-based similarity is that it doesn't take into account that different users have different rating schemes. For instance a user may prefer to rate items that he didn't like at all, while another user prefers to rate items that he liked. To address this issue correlation-based similarity measures are introduced for the case of user similarities. (see equation (3)) This method doesn't directly apply to item similarity since it normalizes values according to the item rating averages which is not much meaningful.

Additionally one other weakness of cosine-based similarity is that it considers all co-purchase of item pairs as a positive effect on their similarity. For instance, a user who has rated i with score 5, and j with score 1 contributes to the similarity value positively where indeed the opposite should be the case. Besides a scenario of giving rating 1 to both items should contribute as much as giving 5 to both items, as it shows the similarity of dislikedness.

3.2.2 *Adjusted Cosine Similarity*

Due to the weaknesses of cosine-based similarity explained in the previous section, we propose a method which both takes into account different user schemes and the effects of dissimilarity and dislikedness similarity. The adjusted cosine-based similarity measure removes these drawbacks by subtracting the corresponding user's average degree from each co-rated pair. This method is first proposed in [20] by *Sarwar et al.* to our best knowledge. Formally, the similarity in this method is given by

$$S_{ij} = \frac{\sum\limits_{u \in S'_{ij}} (r_{ui} - \bar{r_u})(r_{uj} - \bar{r_u})}{\sqrt{\sum\limits_{u \in S'_{ij}} (r_{ui} - \bar{r_u})^2} \sqrt{\sum\limits_{u \in S'_{ij}} (r_{uj} - \bar{r_u})^2}} \quad (12)$$

where $\bar{r_u}$ is the average of the ratings of user u. Experimental results show that this scheme performs slightly better.

3.3 Interpreting Rank Scores

The rank scores $\tilde{R_{ij}}$ obtained at the end of the algorithm are not the actual predictions. They are just numerical values assigned to item-user pairs to sort the items for each user. The obvious usage of these results is identifying the top-N items for each user and recommending them which is realized in [8]. Alternatively these values can be scaled to 1-5 range to answer prediction problem. In our implementation, we linearly scaled up each row of values such that the maximum of each row corresponds to 5.

3.4 Computational Complexity

The bottleneck for the algorithm 3.1 is computing the similarity matrix, S. As both of the similarity measures use the co-rate history of the all item pairs, the computational complexity of computing similarity matrix is $\mathcal{O}\left(m^2 n\right)$. The computation of the rank scores composed of two matrix multiplication and an inverse operation which take significantly less time compared to obtaining similarity matrices.

The beauty of this method is that similarity matrices can be precomputed since they don't evolve quickly. Periodically updating these matrices would be more than enough. Furthermore the value of $\tilde{P}$ which is computed in the eight line of algorithm 3.1 can be precomputed as well. Therefore even though the rating history of a user develops, the actual predictions for a user just require a vector-matrix multiplication which has complexity $\mathcal{O}\left(m^2\right)$. The value of $\tilde{P}$ can also be used as an item-similarity matrix in other item-oriented approaches like [20, 6], which turns the proposed Random Walk Recommender algorithm into a similarity measure method.

4. RESULTS

We conducted an experiment using data from MovieLens[1], an online movie recommendation system, to evaluate the performance of the proposed algorithm. All experiments were performed on an Intel core 2 duo, 2.2Ghz, 2GBytes of memory and a Linux-based operating system.

This data set contains 1,000,209 ratings of 6040 anonymous MovieLens users on 3952 movies. The original data also contains information about user demographics and movie properties, however we didn't use that information since our research is solely on collaborative filtering.

4.1 Experimental Design and Metrics

To evaluate the performance of Random Walk Recommender, we split data into a training and a test set by randomly selecting some percent of the nonzero ratings to be the part of test set and the remaining ones for training. We conducted our experiments in different training set sizes to observe how our algorithm behaves in different sparsity levels. Table 4.1 shows the properties of these datasets. For each dataset, this table shows the ratio of the training set to the whole data and the density of user-item matrix. In addition, the column labeled "Ratings" shows the number of items rated by a user in the corresponding training set, whereas the Cosine-based and Adjusted-Cosine average degrees are the number of links per item the corresponding similarity measurement method explores.

We especially focused on sparse training sets. The first set contains only 0.01 of the whole data with an average of 1.66 item ratings per user. In this set the densities of cosine-based and adjusted cosine-based item-item matrices differ significantly whereas that difference decreases in other training

Training Set Ratio	Density	Average Degrees		
		Rating	Cosine-Based	Adjusted-Cosine
0.01	0.05%	1.66	10.35	4.57
0.05	0.24%	8.28	207.57	118.64
0.1	0.48%	16.58	587.34	395.57
0.15	0.72%	24.75	943.91	701.69
0.2	0.96%	33.19	1262.78	991.35
0.25	1.20%	41.42	1507.13	1229.16
0.6	2.88%	99.30	2435.33	2208.55
0.8	3.84%	132.49	2689.54	2490.33

Table 1: Properties of Training Sets and Similarity Matrices

sets. This is due the fact that adjusted cosine-based method ignores ratings that have average rating value of the user and the ratio of that kind of ratings to the size of training set is high when training set is sparse. The following five training sets were constructed with 0.05 ratio increments. The last two sets were generated to simulate the cases where the data is not sparse.

The quality was measured in three different methods, the first one being the *hit-rate*(HR) which is the ratio of the number of hits to the size of the test set. We refer a predicted rating *hit* if its rounded value is equal to the actual value in the test set. An HR value of 1.0 implies that all the ratings are predicted correctly. One limitation of the hit-rate measure is that it is indifferent to the distance to actual rating in case of a miss. This limitation is addressed by the *Mean Absolute Error, MAE,* which penalizes each miss by the distance to actual rating. The last method is *Root Mean Square Error, RMSE,* a measure that emphasizes large errors compared to *MAE* measure. These measures are formulated below, where n is the number of entries in the test set and P_i and A_i are the predicted and actual ratings of the i^{th} entry, respectively.

(a) $HR = \frac{\#hits}{n}$ (b) $MAE = \frac{\sum_{i=1}^{n} |P_i - A_i|}{n}$ (c) $RMSE = \sqrt{\frac{\sum_{i=1}^{n} (P_i - A_i)^2}{n}}$

Table 2: Evaluation Metrics

4.2 Comparison of Algorithms

We compared our algorithm with a modified *item based top-N algorithm.*[3] As explained in section 3.1, the original Top-N algorithm identifies a ranking of items with a method which is a special case of our algorithm where α equals 0. We modified Top-N algorithm such that it generates ratings based on the ranking it finds. The third method, Default Voting, is an extension to Top-N algorithm where the similarity matrices are computed as if half of the unknown entries are rated with the average rating of the corresponding user. Therefore DV produces very dense similarity matrices regardless of the size of training set. This extension is first proposed in [7] to improve performance where the training data is too small. The results of Random Walk Recommender algorithm are gathered when the parameter α is optimal. Optimal α values are found empirically.

[3]We use abbreviations of Top-N, DV and RWR for item-based top-N algorithm, default voting and random walk recommender respectively.

HR

Alg. \ Ratio		0.01	0.05	0.1	0.15	0.2	0.25	0.6	0.8
Cos	Top-N	0.004	0.08	0.23	0.29	0.29	0.28	0.23	0.23
	DV	0.23	0.23	0.23	0.23	0.23	0.23	0.23	0.23
	RWR	0.08	0.17	0.27	0.32	0.34	0.34	0.29	0.29
Adj. Cos	Top-N	0.002	0.04	0.15	0.25	0.30	0.32	0.34	0.34
	DV	0.22	0.23	0.23	0.23	0.23	0.23	0.23	0.23
	RWR	0.09	0.19	0.28	0.32	0.33	0.33	0.34	0.34

MAE

Alg. \ Ratio		0.01	0.05	0.1	0.15	0.2	0.25	0.6	0.8
Cos	Top-N	3.52	2.41	1.37	1.12	1.10	1.15	1.28	1.30
	DV	1.40	1.40	1.40	1.40	1.40	1.40	1.40	1.40
	RWR	2.40	1.62	1.15	0.98	0.95	0.94	1.10	1.10
Adj. Cos	Top-N	3.55	2.75	1.75	1.26	1.05	0.97	0.95	0.94
	DV	1.41	1.41	1.41	1.41	1.41	1.41	1.41	1.41
	RWR	2.35	1.52	1.12	1.00	0.98	0.96	0.97	0.96

RMSE

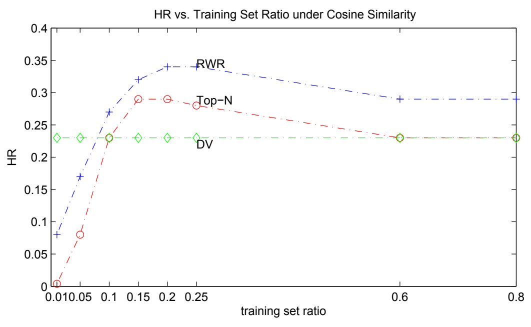
Alg. \ Ratio		0.01	0.05	0.1	0.15	0.2	0.25	0.6	0.8
Cos	Top-N	3.71	2.73	1.73	1.44	1.41	1.46	1.62	1.64
	DV	1.78	1.78	1.78	1.78	1.78	1.77	1.74	1.71
	RWR	2.73	1.94	1.45	1.25	1.08	1.20	1.40	1.41
Adj. Cos	Top-N	3.73	3.02	2.08	1.57	1.32	1.23	1.21	1.20
	DV	1.80	1.80	1.80	1.80	1.80	1.80	1.79	1.79
	RWR	2.68	1.86	1.42	1.27	1.25	1.21	1.23	1.22

Table 3: Experimental Results

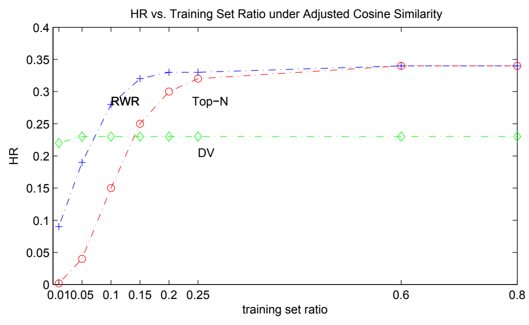
Table 3 shows the results of our experiments. The quality of the predictions increases as HR increases while MAE and RMSE decrease. These metrics behave as parallel to each other in our experiments so we comment only on HR values for the evaluation of algorithms. Random Walk Recommender outperforms Top-N in each of the test cases regardless of the similarity measure used. Default Voting algorithm provides the same quality, 0.23, for all test cases. The first dataset is so sparse that Top-N produce a score very close to 0. Even though the score of RWR is relatively high, it does not perform as well as Default Voting. The optimum α values for the first two test cases are very close to one which makes RWR behave like Pagerank by generating ratings based on popularity. The effect of α is discussed in the following section in detail. When training set is between 10% and 25% of the entire data, RWR becomes able to perform better than DV. In dense datasets Top-N and RWR produces almost the same results (i.e. α becomes 0) under adjusted cosine-based similarity measure. See figure 1 for the comparison of algorithms using different similarity measures.

4.2.1 *Effect of Similarity Measure*

Cosine-based similarity measure reveals more item relationships compared to adjusted cosine-based measure since the latter one ignores the ratings that are equal to the user's average rating. Adjusted measure interprets these ratings as if the user is indifferent to the them. This fact reduces the density of similarity matrix more as the training data gets smaller. Therefore cosine-based measure produces better predictions in extremely sparse training sets. However, as the adjusted measure has more and more training data it learns item similarities better than the cosine-based measure due to the reasons discussed in section 3.2. The graph of optimal α values in figure 2 also indicates that it is al-



(a) Cosine



(b) Adjusted Cosine

Figure 1: Comparison of Algorithms

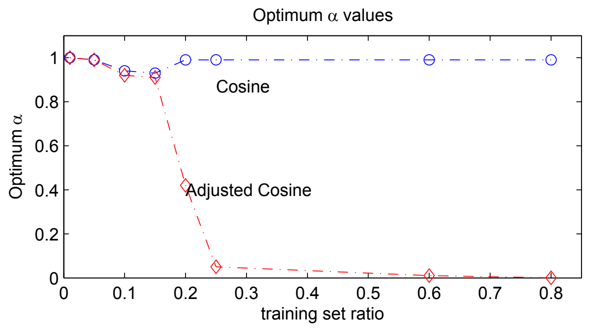


Figure 2: Optimal α Values

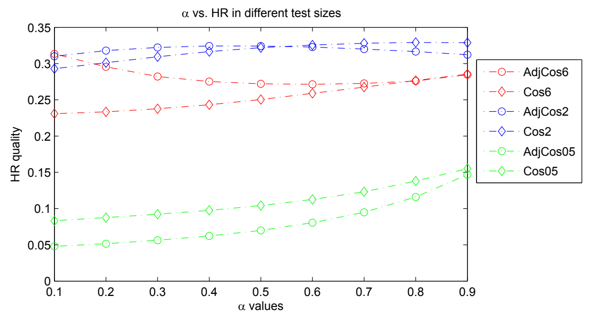


Figure 3: Effect of α in different test sizes

ways possible to improve the similarity matrix obtained by the cosine-based measure since α is close to 1 in all test cases whereas the adjusted cosine-based measure becomes optimal since α approaches 0 as training set gets bigger. In the overall, Random Walk Recommender algorithm combined with the adjusted-cosine based measure outperforms the other combinations.

4.2.2 *Effect of the Parameter α*

The parameter α determines the expected length of the random walk. There is a direct correlation with the length of the random walk and the transitive associations identified by Random Walk Recommender method. If the similarity matrix is good enough to describe relationships between items without necessitating transitive associations, the length of the walk should be close to 0. This case usually occurs in dense data sets. However when the training data is not dense enough, similarity measures need to use transitive association to identify actual similarities. Depending on the sparseness of the training data, a length of 2 to 10 for the random walk would be enough to explore relationships between items. When it comes to very sparse data sets, where similarity measures fail to find meaningful relationships, the length of the walk could be extremely large. As the length of the walk increase, Random Walk Recommender tends to produce ratings according to the popularity of items regardless of the user. This is analogous to Pagerank where allowing an infinite length walk produces a ranking independent of the initial distribution.

Figure 3 shows the α vs. prediction quality graph of RWR in three different training data sets. The dense data set has the training set ratio of 0.6. In this case, RWR gives best predictions when α is close to 0 under adjusted-cosine measure and the predictions get worse as α increases. On the contrary, RWR under cosine-based measure surprisingly produce better predictions as α increases. Therefore we can conclude that adjusted measure learns the data better. The sparse data set has the training set ratio of 0.2. α takes its optimal value around 0.42. This result confirms our expectation that in reasonably sparse data sets, determining an average length of 2 to 10 for RWR gives best results. Figure 2 also shows that the reasonably sparse training sets have optimal α values between 0.92 and 0.42. In extremely sparse data sets where similarity measures identify very few dependencies, the optimal α values are very close to 1 and Random Walk Recommender performs significantly better than Top-N.

5. DISCUSSION AND CONCLUSIONS

In this paper, we presented and experimentally evaluated a model-based item-oriented collaborative filtering algorithm. Our results showed that Random Walk Recommender algorithm outperforms a slightly modified version of item based top-N algorithm in all test cases since top-N is a special case of Random Walk Recommender. The proposed algorithm performs significantly better than top-N algorithm especially when training data is sparse. Furthermore, experiments show that optimal α values can be predicted in advance according to the density of training data. Therefore most of the computations can be pre-computed which makes the algorithm have the same computational complexity with top-N algorithm. For extremely sparse data sets optimal α values approaches 1 whereas it approaches to 0 as data gets denser. It takes values between 0.2 and 0.9 in reasonably sparse data sets where Random Walk Recommender captures some transitive associations between items.

One important contribution of this paper is that it gives rise to enhancing similarity matrices under sparse training data in which typical similarity measures fail to discover all item relationships. These similarity matrices can be used in typical item-oriented approaches and their performance can be compared to user-oriented approaches in a future work.

Lastly, the experiments reveal that adjusted cosine-based similarity measure learns the relationships as training data gets dense. Whereas in extremely sparse data sets, cosine-based similarity measure is able to capture more relationships. However Random Walk Recommender combined with the adjusted cosine-based similarity outperforms other combinations in all test cases.

6. ACKNOWLEDGEMENTS

We would like to thank GroupLens research group of University of Minnesota for providing valuable MovieLens data.

7. REFERENCES

[1] Movielens data set. Available at `http://www.grouplens.org/node/73#attachments`, Nov. 2007.

[2] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, 2005.

[3] B. Awerbuch, B. Patt-Shamir, D. Peleg, and M. Tuttle. Improved recommendation systems. In *SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1174–1183, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics.

[4] Y. Azar, A. Fiat, A. Karlin, F. McSherry, and J. Saia. Spectral analysis of data. In *STOC '01: Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 619–626, New York, NY, USA, 2001. ACM.

[5] J. K. Badrul Sarwar, George Karypis and J. Riedl. Application of dimensionality reduction in recommender system a case study. In *WebKDD-2000 Workshop*, 2000.

[6] R. Bell, Y. Koren, and C. Volinsky. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 95–104, New York, NY, USA, 2007. ACM.

[7] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. pages 43–52, 1998.

[8] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, 2004.

[9] P. Drineas, I. Kerenidis, and P. Raghavan. Competitive recommendation systems. In *STOC '02: Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 82–90, New York, NY, USA, 2002. ACM Press.

[10] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12):61–70, 1992.

[11] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.

[12] Z. Huang, W. Chung, T.-H. Ong, and H. Chen. A graph-based recommender system for digital library. In *JCDL '02: Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*, pages 65–73, New York, NY, USA, 2002. ACM.

[13] Z. D. Huang, Z. and H. Chen. A link analysis approach to recommendation with sparse data. In *Americas Conference on Information Systems*, New York, NY, 2004.

[14] J. Kleinberg and M. Sandler. Convergent algorithms for collaborative filtering. In *EC '03: Proceedings of the 4th ACM conference on Electronic commerce*, pages 1–10, New York, NY, USA, 2003. ACM.

[15] J. Kleinberg and M. Sandler. Using mixture models for collaborative filtering. In *STOC '04: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 569–578, New York, NY, USA, 2004. ACM.

[16] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Recommendation systems: A probabilistic analysis. *focs*, 00:664, 1998.

[17] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.

[18] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[19] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, New York, NY, USA, 1994. ACM.

[20] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 285–295, New York, NY, USA, 2001. ACM.

[21] U. Shardanand and P. Maes. Social information filtering: algorithms for automating word of mouth. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 210–217, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[22] L. Ungar and D. Foster. Clustering methods for collaborative filtering. In *Proceedings of the Workshop on Recommendation Systems.* AAAI Press, Menlo Park California, 1998.