Harnessing the Power of "Favorites" Lists for Recommendation Systems

by

Maryam Khezrzadeh
B.Sc., Sharif University of Technology, 2007

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

© Maryam Khezrzasdeh, 2009
University of Victoria

All rights reserved. This dissertation may not be reproduced in whole or in part, by photocopying
or other means, without the permission of the author.

Harnessing the Power of "Favorites" Lists for Recommendation Systems

by

Maryam Khezrzadeh
B.Sc., Sharif University of Technology, 2007

Supervisory Committee

Dr. Alex Thomo, Supervisor
(Department of Computer Science)

Dr. William W. Wadge, Supervisor
(Department of Computer Science)

Supervisory Committee

Dr. Alex Thomo, Supervisor
(Department of Computer Science)

Dr. William W. Wadge, Supervisor
(Department of Computer Science)

ABSTRACT

This thesis proposes a novel recommendation approach to take advantage of the information available in user-created lists. Our approach assumes associations among any two items appearing in a list together. We consider two different ways to calculate the strength of item-item associations: frequency of co-occurrence, and sum of Bayesian ratings (SBR) of all lists containing the item pair. The latter takes into consideration not only the number of lists the items have co-appeared in, but also the quality of the lists. We collected a data set of user ratings for books along with Listmania lists on Amazon.com using Amazon Web Services (AWS). Our method shows superior performance to existing user-based and item-based collaborative filtering approaches according to the resulted Mean Absolute Error (MAE), coverage, precision and recall.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Problem Definition and Motivation

The massive increase in volume of online information makes it hard to find new relevant items, such as books, music, movies, journal articles, etc. This increase is due to the huge shift from physical media to digital media. Recommender systems try to help users by finding items that may fit their needs and interests, from books, news, music and movies to restaurants, holiday destinations and even friends.

There is a strong commercial interest in recommendations systems according to the reports published by online retailer and entertainment web sites. Amazon claims that 35% of its product sales result from recommendations while Netflix "retains" users by providing interesting movie recommendations. 2/3 of the total movies rented from Netflix were recommended to the user [1]. From the user's perspective, recommendation systems save a lot of time and effort, reduce confusion and are most commonly a source of new discoveries which they were otherwise unlikely to achieve.

1.2 Background

Over the years, various approaches to build recommendation systems have been developed and new algorithms and applications are emerging everyday. Content-Based (CB) filtering [2] as an early approach to produce recommendations is based on correlations between the content of items and user preferences. This method relies on the machine analysis of contents and therefore entails the items to be of a machine-parsable format (e.g. text) or some attributes should be assigned to the items man-

ually. Due to limitation of resources in the early days of content-based systems, non-text items such as art work and music could not be analyzed effectively and that limited the practical usage of CB systems.

The collaborative Filtering (CF) method [2, 3, 4, 5] which drives on historical information helps people make choices based on opinions of other like-minded people. It draws on a heuristic idea: people who agreed in the past are likely to agree in the future. It is one of the most successful approaches to develop recommendation systems. Primarily, CF was mostly accepted because it overcame some of the major limitations of Content-Based filtering systems [2]. Firstly, since CF uses the user preferences (likes and dislikes) for items, it can be easily applied to any type of items and secondly, CF algorithms are capable of finding serendipitous recommendations.

As one can notice, the crucial element of a recommendation system is the notion of user preferences for items which is commonly referred to as user-item interaction. In other words, we have to know what the user likes (and probably what she does not like) in order to make recommendations. User-item interactions may be represented as a two dimensional matrix where a value in row i and column j of the matrix shows the preference of user i for item j. Collaborative filtering recommendation algorithms find similar users based on their previous interactions with items. The like-minded (i.e. similar) neighbors of a user directly contribute to the process of finding recommendation for that user.

The classic CF approach, which is also known as user-based method, has some serious scalability and quality challenges associated with it [6]. These challenges have led to the design of a similar item-based scheme which utilizes item-item similarities rather than user-user similarities. The item-based collaborative filtering recommendation algorithms are claimed to address these two challenges simultaneously leading to the design of more accurate and more scalable recommender systems [7].

While the item-based recommendation algorithm alleviates some of the scalability issues with user-based approach, they both suffer from common problems, such as *sparsity* (i.e. lack of enough information in user-item interaction matrix) and *non-transitive item associations* where the system is not smart enough to derive item-item or user-user associations through transitivity.

To alleviate these problems, some alternative model-based recommender algorithms are explored which mainly aim at augmenting item ratings resulting in a denser user-item interaction matrix or seeking alternative ways to drive item-item or user-user relationships. Dimensionality reduction techniques and more specifically

Singular-value decomposition [8], have been applied in the context of recommender systems to capture the similarity between users and items from a condensed and less sparse interaction matrix. Latent Semantic Analysis [9, 10] finds the patterns of interaction between users and items. The discovered model is then used to predict the degree a user will like an item.

Several graph-based [11] approaches have been explored which adopt the association information retrieval [12] and link analysis [13] techniques along with the exploration of transitive associations between users and items to address different issues with recommender systems. Other data mining techniques like classification [14], clustering [15] and association rule mining [12, 16, 17] have also been applied to recommender systems some of which incorporate the content information as well as preference information to improve the quality of recommendations.

While applying different algorithms and techniques to the basic user-item interaction matrix results in more elaborate and successful recommender systems, the type of available information in applications that use recommendations has greatly affected the way new systems are designed. For example the existence of trust data in social networks has inspired the design and development of trust-based recommendation systems [18]. Therefore, as new sources of information are made available, new recommendation approaches can be designed to utilize this information.

1.3 The Proposed Solution

Nowadays, many online e-commerce and entertaining recommender systems, enable the users to create and manage lists of their favorite items. Many online music services like Last.fm[1], iLike[2] and Pandora[3] allow the users to create custom radio stations and playlists from any of the audio tracks in their music library. Amazon[4], the biggest online retailer in the US, recently has offered the Listmania list creation service where users can put together a list of related books, DVDs, music, etc. MovieLens[5], Netflix[6], YouTube[7] and many more are other examples of websites offering such services.

[1] http://www.last.fm/
[2] http://www.ilike.com/
[3] http://www.pandora.com/
[4] http://www.amazon.com
[5] http://www.movielens.org/
[6] http://www.netflix.com/
[7] http://www.youtube.com/

The effort a user puts into creating a list is of great importance to the task of recommendation. That is because, unlike user profiles, which are currently used in many recommendation services, user-created favorites lists usually have a unique theme, topic and taste. Therefore, we can consider each favorite list to represent a collection of highly related items. Moreover, these relationships are typically approved by two groups of human experts: the user who creates the list and the viewers who vote for the list. A reviewer, signifies a list as helpful by voting *yes* to the list and voting *no* in case the list is not appealing as a related set of items. Incorporating the opinion of voters has significant benefits towards distinguishing strong item associations versus weak, limited or unreasonable item-item relationships.

This thesis offers a novel, efficient and flexible way of extracting item-item relationships out of these lists to enhance the quality of recommendations generated for users in such systems. We call this new approach, the Collective Intelligence ReCommendation (CIRC), since it draws upon the collective intelligence of the users creating cohesive collections as well as the opinion of the lists' viewers. The experimental results show that the research presented by this thesis has successfully revealed a new potential to extract item-item associations from valuable and commonly overlooked information in user-created lists. CIRC outperforms user-based and item-based CF methods according to the resulted F-Measure, Mean Absolute Error and coverage. The results of this research have been published in ACM RecSys 2009 [19].

1.4 Agenda

The material presented above in this chapter briefly states the problem of making recommendations, it's importance and how well it has been studied recently. It gives a sketch of the new approach to tackle the problem and claims that CIRC works better than common approaches when extra information is available. The rest of this thesis is organized as follows.

Chapter 2 describes in details the problem which is to be tackled along with its context, its impact and the overall motivation for the use of this list-based model to produce recommendations

Chapter 3 introduces CIRC, the new Collective Intelligence ReCommendation system. It gives the preference model that CIRC operates on to imply a weighted item-item graph. Two ways of calculating edge weights to reflect the degree

of associations between two items are introduced in this chapter. Finally the recommendation algorithm used by CIRC is given.

Chapter 4 describes the data set used in this research along with data collection procedure. It also presents the experimental settings to evaluate the performance of CIRC.

Chapter 5 reports the results of several experiments to show the effectiveness of CIRC over item-based and user-based collaborative filtering approaches

Chapter 6 contains a restatement of the claims and results of the dissertation. It also enumerates avenues of future work for further development of the concept and its applications.

Chapter 2

Problem Desciption

This chapter presents an overview of the recommendation systems. It describes the state-of-the-art in recommendation systems as well as various limitations of the current recommendation methods. The purpose of this chapter is to introduce the problem of producing recommendations and to motivate the reader about this study. A taxonomy will be given at the end of this chapter witch places this research and its particular special features within the context of the overall area.

2.1 Why Are We Interested In Recommender Systems?

The term Collaborative Filtering (CF) was first introduced in the mid-1990s [2, 4, 5] to describe the first implementations of CF recommender systems. Since then, there has been much work done both in industry and academia to develop new approaches for recommender systems. However, there is still an increasing interest in the field because there are lots of opportunities to improve the current practice and also because of the abundance of the real world applications that help users deal with the issue of information overload by providing personalized recommendations. Information overload results from the huge shift from physical media to digital media which greatly increases the volume of online content. Besides research value and practical usage of recommendation systems, there is also an absolute commercial interest in these systems according to the reports published by online retailer and entertainment web sites.

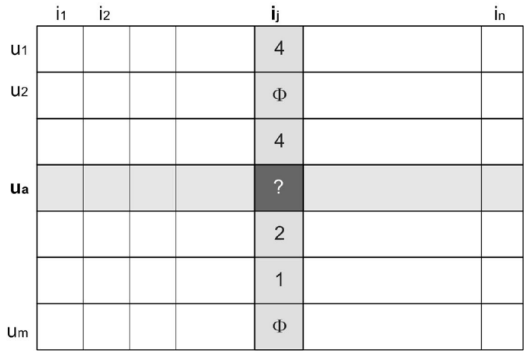

Figure 2.1: User-Item Interaction Matrix

2.2 Formalization of the Recommendation Problem

Recommender systems turned into automated methods of information filtering in mid-1990s when researchers focused on developing rating-based recommendation approaches. In its simplest formulation, the task of recommendation is reduced to a prediction problem where the goal is to predict ratings for the items yet unseen by the user. This prediction of ratings is usually based on the previous ratings that the user has given to the items. Once the ratings for unseen items are predicted, the problem is simply to recommend items to the user which have higher estimated ratings.

Formally we formulate the recommendation problem as follows: Let C be the set of all *users* (Customers) and P be the set of all *items* (Products) such as books, DVDs, music, etc. Let u be a *utility function* that measures the usefulness of item p to user c [20], i.e., $u : C \times P \rightarrow R$ where R is a nonnegative integer or real number within a certain range. The utility of an item may be expressed either *explicitly* or *implicitly*. Ratings, purchase history and relevance feedback are examples of explicit utility defined by user. Implicit utility on the other hand is obtained by monitoring user activities and behaviors. For example in an online music recommendation website, user's actions like listening, playing, stopping and skipping music may be utilized to gain implicit utility values. In general, utility can be an arbitrary function, however, in recommendation systems it's usually represented by a rating. Figure 2.1 illustrates an example of a user-item interaction matrix.

Based on formal definitions mentioned above the main problem of recommender

systems is that the utility u is usually not defined on the whole space $C \times P$. Note that in today's systems the items space can be very large in some applications. As a result, a user c may express her opinion about a subset of the total items in item space, not all of the items. Therefore, u needs to be extended (extrapolated) to the whole user-item space. Once this extrapolation of utility function is done, the actual task of recommendation to a user is to select the highest rated items among all the items with predicted ratings.

There are many different ways to predict the ratings for the not-yet-rated items. In fact, recommender systems are classified according to their approach to estimate ratings. There are three classes (i.e. categories) of recommendation systems: Content-based Filtering, Collaborative Filtering and Hybrid Methods. Each of these methods will be addressed in one of the subsequent sections to complete the big picture of recommendation systems.

2.3 Recommendation Algorithms

2.3.1 Content-Based Methods

In a content-based recommendation system the predicted rating for item p according to user c is computed based on the *similarity* between the content of item p and that of other items previously rated by c. For example in a music recommendation application, the new music to be recommended to the user, will be checked against other musics that the user has liked in the past to find commonalities like rhythm, tonality, timbre, tags, etc. Then musics with higher degrees of similarity to user's preferences will be recommended.

To compute the similarity between items, we first need to describe the items. Each element of the item space has a *profile*, i.e., a set of properties. These attributes (i.e. features, properties), are usually extracted from item's content. Therefore, for a content-based method to work, the item content should be of a machine-parsable format like text. As a result, many current content-based systems focus on recommending items containing textual information, such as documents and news. However, it is possible for an application to have non-textual items which are surrounded by a halo of text. For example, music as a non-textual item may be described using its available textual information like expert-applied metadata, reviews, playlists, lyrics and associated tags. Some of this information is general enough to be utilized in

describing almost any other type of items. For example reviews, and associated tags can be applied to movies, restaurants and many other non-textual items.

The content of items in the systems mentioned above is usually described with *keywords*. To select the best keywords to describe an item, we need a measure to specify the "importance" (or "informativeness") of a word related to an item. Assume that the halo of text surrounding an item can be represented by a text document d_j. One of the best-known metrics to measure the importance of keyword k_i in document d_j is the *Term Frequency-Inverse Document Frequency* measure (TF-IDF) defined as follows:

TF-Term Frequency Let f_{ij} be the frequency of keyword k_i in document d_j. Term frequency or normalized frequency of keyword k_i in document d_j is defined as:

$$tf_{ij} = \frac{f_{ij}}{\sum_z f_{zj}}$$

IDF-Inverse Document Frequency Assume that N is the total number of documents and that n_i is the number of documents containing keyword k_i. Inverse document frequency of keyword i is then defined as:

$$idf_i = log\frac{N}{n_i}$$

TF-IDF To measure the importance of keywords in documents we penalize the keywords that appear in many documents because these keywords are not as informative and are not useful in distinguishing between a relevant and a non-relevant document. The importance (weight) of keyword k_i in document d_j is computed as:

$$w_{ij} = tf_{ij} \times idf_i$$

Intuitively, the weight increases proportionally to the number of times a word appears in a document but is offset by the frequency of the word in all documents (i.e. collection or corpus). Inverse document frequency factor diminishes the weight of terms that occure very frequently in the collection (like the term "the") and increases the weight of the terms that occur *rarely*.

Based on the definitions above, we represent the content of an item as follows:

$$Content(d_j) = (w_{1j}, \ldots, w_{mj}) \tag{2.1}$$

where $(k_1, \ldots, k_m)$ are the most *important* keywords in document d_j. $(k_1, \ldots, k_m)$ defines the vector space in which document contents are described

As is was stated earlier, in a content-based recommendation approach, items with higher degree of similarity to user preferences are recommended to the user. We defined the content of items in equation 2.1. To find the similarity of an un-rated item to user profile (i.e. user preferences), one way is to represent the user profile in the same vector space as the item content. In this way we may use vector similarity measures like cosine similarity to predict how much a user will like an un-seen item.

More formally, let $Profile(c) = (w_{1c}, \ldots, w_{kc})$ where each w_{ic} represents how important keyword k_i is to user c. w_{ic} may be calculated from the content vectors of the items that he has liked in the past using various methods [21, 22]. Using cosine similarity, we define the utility function as:

$$u(c, p) = cos(\vec{w_c}, \vec{w_p}) = \frac{\vec{w_c}.\vec{w_p}}{||\vec{w_c}||_2 \times ||\vec{w_p}||_2} \tag{2.2}$$

where $\vec{w_c}$ and $\vec{w_p}$ are TF-IDF weight vectors for user c and item p respectively.

Besides the cosine similarity described in equation 2.3.1 there are also other methods devised for content-based recommendation. Bayesian classifiers and various machine learning techniques like clustering, decision trees and artificial neural networks are examples of such methods. These methods learn a model from the user preferences and behaviors and use that model to classify new items into relevant-irrelevant items or predict how much a user will like them.

There are known limitations associated with content-based recommendation systems discussed in the rest of this section.

Limited Content Analysis

Content-based recommendation has benefited a lot from text retrieval techniques. While these techniques perform well at extracting features from documents, feature extraction methods from other type of data like audio and video streams and images are still in their early stages. Due to the lack of successful feature extraction techniques for non-text data, manual annotation by human experts has been examined in various applications like Pandora and Soundflavor[1] music recommendation systems. Practically, human content analysis does not scale. Therefore, the usage of content-

[1] http://soundflavor.com/

based recommendation is limited by the type of items an application recommends to its users.

Other limitation of CB recommendation systems is that these systems are not capable of distinguishing between good and bad items if the two items have the same features describing their content. For example two Java programming books more probably use the same set of words and their content vectors (i.e. the TF-IDF weight vectors) are very similar. Assume that one of these books is a well written book extensively used and respected by the professional community while the other book is poorly written. A content-based method can not distinguish between these two books and can't prefer the first book over the other one at recommendation time.

Overfitting

Making serendipitous recommendations is usually a desirable feature in recommendation system. Keep in mind that a recommendation is more useful when it reveals non-trivial items of interest to a user which will otherwise be impossible or labor intensive for him to find. A content-based system however, tends to find the most similar items to the ones already seen by the user. This approach may some times be very undesirable, such as recommending an alternative news page describing the same news or recommending all movies by Quentin Tarantino to a user that once liked one of his movies.

Cold-Start Users

A user should rate a sufficient number of items before the content-based system is able to make recommendations for her. For a new user with few rated items, the system can not produce accurate recommendations because the user preferences can't be fully understood.

2.3.2 Collaborative Filtering

Unlike content-based recommendation methods, Collaborative Filtering (CF) methods help people make choices based on opinions of *other like-minded people*. It is one of the most successful approaches to develop recommendation systems. Primarily CF systems, also known as *social information filtering* systems, were mostly accepted because they overcame some of the major limitations of content-based filtering systems which were mentioned before.

The term *collaborative filtering* was first introduced by Goldberg et al. [3], to describe Tapestry - one of the first implementations of CF recommender systems. Tapestry relies on each individual knowing others so that he can request for documents based on opinions of like-minded people in his community. However, in real word situations, social information filtering can't depend on people knowing each other. Video Recommender [23], Ringo [2] and GroupLens [4, 5] are the first systems developed shortly after to *automate* prediction by using collaborative filtering algorithms.

There are three general approaches to collaborative recommendation: item-based, user-based and model-based. Each of these collaborative filtering methods will be discussed in more details in the following sections.

User-based Approach

In a user-based recommendation system, "peers" of a target user c, i.e., other users with similar tastes, are found, and the items that are most liked by the "peers" are recommended to the user . For example, in a music recommendation system, users who have similar listening habits are considered as peers (or neighbors) and in a movie recommendation system users who have rated the same movies similarly are peer users.

As it was mentioned in section 2.2 , the collection of previously rated items by the users may be represented as a user-item interaction matrix. For user c and item p, user-based collaborative filtering systems make rating prediction r_{cp} based on the user-item interaction matrix as follows:

1. The algorithm finds the set $Neighbours(c)$ of N users that are most similar to user c and who have rated item p. N can be any number between 1 and the total number of users.

2. The ratings of most similar users (neighbors) are aggregated to compute r_{cp}

User-based recommendation algorithms differ in the way they compute neighbors and in the aggregation function they use. The simplest aggregation function is the average rating given to item p by the neighbors of user c as defined in the following equation.

$$r_{cp} = \frac{1}{N} \sum_{c' \in Neighbours(c)} r_{c'p} \tag{2.3}$$

However, the more similar users c and c' are, the more c' should participate in the prediction of r_{cp}. *Weighted sum* is an aggregate function which implements this idea by using $sim(c, c')$ as a weight and is defined in equation 2.4. Multiplier k is a normalizing factor and is defined in equation 2.5.

$$r_{cp} = k \sum_{c' \in Neighbours(c)} sim(c, c') \times r_{c'p} \tag{2.4}$$

$$k = \frac{1}{\sum_{c' \in Neighbours(c)} sim(c, c')}. \tag{2.5}$$

While weighted sum has some nice properties, it does not consider the fact that different users may use the rating scale differently. For example Mary in average gives higher ratings to all the items she rates compared to John. To address this problem, the *adjusted* weighted sum (see equation 2.6) is defined and commonly used in user-based recommendation systems. In this approach the deviation of ratings from the average rating is used as opposed to the absolute rating used in weighted sum mentioned earlier. k in equation 2.6 is the normalizing factor defined in 2.5.

$$r_{cp} = \bar{r}_c + k \sum_{c' \in Neighbours(c)} sim(c, c') \times (r_{c'p} - \bar{r}_{c'}) \tag{2.6}$$

Various approaches have been used to compute the similarity $sim(c, c')$ between users based on the ratings of items rated by both users. Let $P_{cc'} = \{p \in P | r_{cp} \neq \phi \text{ and } r_{c'p} \neq \phi\}$ be the set of items rated by both users c and c'. Using the *Pearson correlation coefficient* [24, 7] we define user similarity as:

$$sim(c, c') = \frac{\sum_{p \in P_{cc'}} (r_{cp} - \bar{r}_c)(r_{c'p} - \bar{r}_{c'})}{\sqrt{\sum_{p \in P_{cc'}} (r_{cp} - \bar{r}_c)^2 \sum_{p \in P_{cc'}} (r_{c'p} - \bar{r}_{c'})^2}} \tag{2.7}$$

Let's define user vectors $\vec{c}$ and $\vec{c'}$ in n-dimensional space where $n = |P_{cc'}|$. In cosine-based approach [24, 7], the similarity between two users is defined as the cosine of the angle between two user vectors:

$$sim(c, c') = \cos(\vec{c}, \vec{c'}) = \frac{\vec{c} \cdot \vec{c'}}{||\vec{c}||_2 \times ||\vec{c'}||_2} \tag{2.8}$$

User-based collaborative filtering has been promising in both research and practice, however, there are two fundamental challenges associated with it (see [10]).

Scalability is the first challenge. Today's systems with high number of users demand the CF algorithms to search millions of potential neighbors in real-time to make predictions for a target user. In other words the computational complexity of these systems grows linearly with the number of users. Further, the high dimensionality of user preferences can cause serious scale problems. Consider a CF system which stores the browsing pattern of users as implicit user-item interaction. For a frequent visitor the algorithm needs to deal with a long user row which dramatically slows down the number of neighbors that can be searched per time unit. Since in most of the applications the recommendations are based on the target user's current basket, the system can't benefit from pre-computed user similarities [6]. The throughput, or the number of target users the system can serve per time, could be increased by increasing the number of recommender servers; however, this technique does not decrease the latency of each recommendation task and therefore is not helpful in real-time applications [6].

The second challenge that is also a challenge for every recommender system is to improve the *quality* of the recommendations. The quality of a user-based collaborative filtering algorithm is in conflict with its scalability. To improve the performance of user-based CF systems, one way is to cluster the users and limit the nearest-neighbor search task among the users in target user's cluster [24]. The nature of user-based CF systems however, implies that the less time is spent searching for neighbors, the more scalable the algorithm will be and the worse its quality. These approaches speed up the algorithm significantly while reducing the quality of recommendations.

Item-based collaborative filtering recommendation algorithms are claimed to address these two challenges simultaneously leading to the design of more accurate and more scalable recommender systems [10].

Item-based approach

The item-based algorithm is similar to user-based algorithm except that it computes the similarity between *items* instead. The predicted rating of items for users are then gained form item similarities. Different similarity measures have been examined in designing item-based algorithms. Cosine, adjusted cosine and Pearson correlation are examples of measures which could be applied to user-item interaction matrix [7, 6]. $u(c, p)$ predicts the interest of target user c in item p based on her profile. The more similar the target item is to the items previously loved by the target user, the more

likely the user will be interested in that item in the future and hence the higher the utility (see equation 2.9).

$$r_{cp} = \frac{\sum_{p'} sim(p, p') r_{cp'}}{\sum_{p'} sim(p, p')} \tag{2.9}$$

An experimental study has shown that item-based algorithms are more efficient and provide comparable or better recommendation quality [6].

Model-based approach

User-based and item-based collaborative filtering methods are *memory-based* approaches since they consider the entire user-item interaction matrix to predict ratings of unseen items. In contrast, *model-based* collaborative filtering algorithms *learn* a model from the collection of ratings and make predictions based on this model. Model based methods alleviate some of the problems associated with memory based collaborative filtering methods by augmenting item ratings resulting in a denser user-item interaction matrix or seeking alternative ways to drive item-item or user-user relationships. More specifically model-based methods have been shown to be effective in addressing the *sparsity* and *non-transitive item associations* problems:

Sparsity : In a given database of user-item interactions, usually the number of items a user has rated, purchased or reviewed is very small compared to the total number of items in the domain. Similarly, most of the items are rated, purchased, or observed by a few users. Thus, producing recommendations for a user with a short list of ratings may be difficult as well as recommending items that have not received sufficient ratings. The *cold-start* problem is an extreme case of the sparsity problem when a user does not receive any recommendations or an item is never recommended.

Non-transitive item associations if two similar items have never been rated by the same user, no association between the items can be derived.

Dimensionality reduction techniques, such as singular-value decomposition (an algebraic feature extraction technique) [8], have been applied in the context of recommender systems to capture the similarity between users and items from a condensed and less sparse interaction matrix. Latent semantic analysis [9, 10, 25] finds the patterns of interaction between users and items. The discovered model is then used to predict the degree a user will like an item.

Several graph-based [11] approaches have been explored which adopt the association information retrieval [12] and link analysis [13] techniques along with the exploration of transitive associations between users and items to address different issues with recommender systems. Other data mining techniques like classification [14], clustering [15] and association rule mining [12, 16, 17] have also been applied to recommender systems some of which incorporate the content information as well as preference information to improve the quality of recommendations. The idea of incorporating content information in collaborative filtering methods is followed in the third category of recommender systems which is called *hybrid* methods. In the next section we review various approaches to hybrid recommendation systems.

2.3.3 Hybrid Methods

Hybrid approaches combine content-based and collaborative methods to avoid certain limitations associated with each of these methods individually. Several ways to combine content-based and collaborative approaches are studied by research community [26, 14, 27, 28, 29, 30, 15] and may be classified as follows:

1. **Combining the output (rating predictions) from both methods.** In this case, we need to have two separate implementations, one for each recommendation algorithm. Having two predicted rating, we may combine them into a single rating value using either a linear combination [27] or a voting scheme [28]. We can also alternate between the two methods at any time considering which one gives the better prediction satisfying a quality criteria. For example, we may choose the method which has higher confidence in its recommendation [31] or we can choose the one which bears the rating that is more consistent with previous ratings of the target user [32].

2. **Cascading collaborative characteristics into a content-based approach or vice versa**. Examples of systems using this hybrid approach include systems which maintain content-based profile of the users as well as the pure ratings used in the collaborative approach. By augmenting the user-item interaction matrix in this way, a user receives recommendations both for items which are scored high by similar users or items as well as items which score high against user profile. On the other hand, its possible to add collaborative filtering characteristics to content-based method. For example, we can group the content-based

user profiles using dimensionality reduction techniques to create a collaborative view for each user group [30].

3. Incorporating both content-based and collaborative characteristics into a single model like the ones referred to in section 2.3.2, model-based approach.

2.4 What Is This Thesis About?

As was discussed above, over the years various recommendation approaches has been introduced which utilize different techniques from machine learning, information retrieval, stochastic and probability methods. While current methods of recommendation have performed well in several applications, there is still room for significant extensions so that recommendation systems can provide better recommendations and also be able to provide recommendations in more complex contexts such as recommending vacation destinations and financial advice.

We reviewed three main categories of recommendation approaches: content-based, collaborative filtering and hybrid methods. All these methods rely on the user and item profiles and the history of interaction between users and items of a system in order to compute user-user and item-item similarities and to predict rating for unrated items. However, new sources of information are emerging every day which leaves us with this question: can we use these information to provide better recommendations or apply recommendation systems in new applications and extended contexts? Examples of such information include data about the interaction between users in social networks which in some cases also contains trust information. Availability of such data has inspired researchers to seek new methods of computing user-user and item-item similarities based on social connections and interactions. Also, there have been new ways of predicting ratings based on trust data.

Another example of emerging new information is the user-created collections of items. Music and video playlists, wishlists and book collections are all examples of user-created *lists*. The effort a user puts into creating a list is of great importance to the task of recommendation. We can consider each list representing a collection of highly related items. In this thesis we try to investigate ways of utilizing valuable information provided by user lists to infer item-item similarities. The new recommendation method presented in this thesis is not classified as any of the three categories defined earlier in this chapter. We call it *Collective Intelligence ReCommendation*

(CIRC) since it draws upon the intelligence of the users creating cohesive collections and sometimes the opinion of the lists' viewers. Figure 2.2 shows a taxonomy of different recommendation approaches including CIRC, our new proposed recommendation approach. CIRC offers some advantages which are inherent in its unique way of inferring item-item similarities.

1. Unlike most of the recommendation systems which suffer from the cold-start problem, our system does not require a user to rate a significant number of items in order to expect high quality recommendations. A user just needs to express her interest in one item to be eligible to receive recommendations. Furthermore, the new items (the so called "cold-start" items) are guaranteed to be recommended if they appear in at least one list. Moreover, most recommendation algorithms are either specifically designed to deal with cold-start situations or they use separate algorithms for making cold-start recommendations; CIRC, on the other hand, similarly uses the same algorithm for both regular and cold-start recommendations.

2. The list-based model is built off-line and is kept up to date at negligible cost. As a result, our approach offers a fast, affordable, and scalable solution to real-time web applications seeking recommendations for their ever-growing number of users.

We will show later in this thesis how CIRC outperforms commonly used user-based and item-based collaborative filtering recommendation approaches.

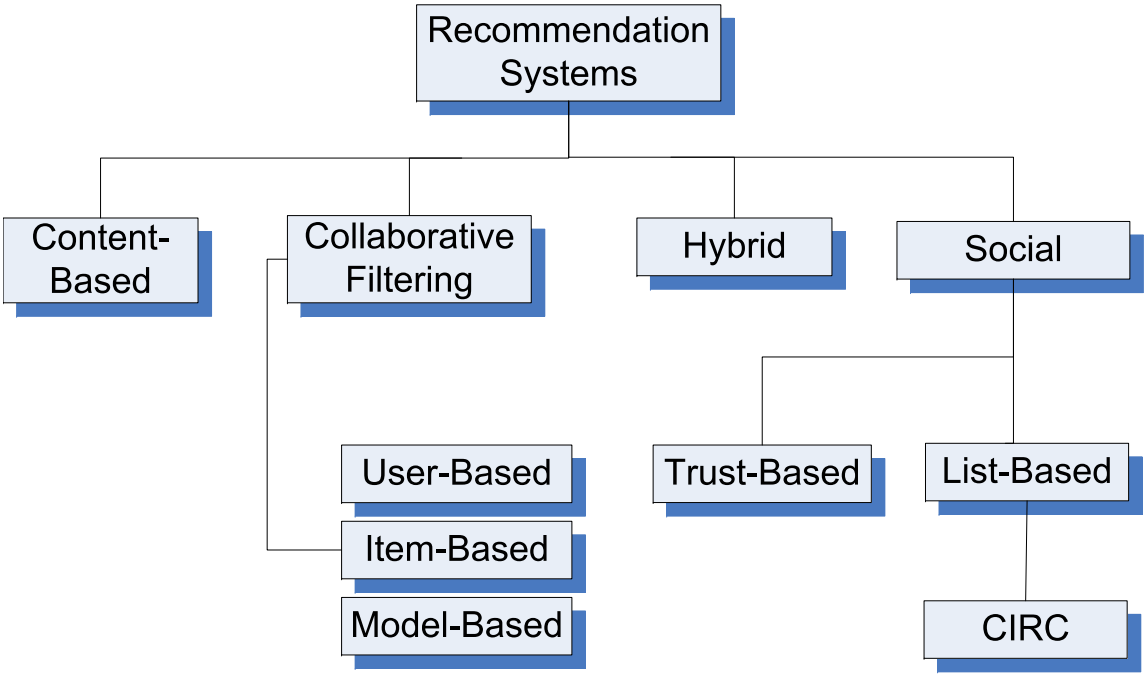

Figure 2.2: Classification of recommendation systems

Chapter 3

CIRC

In this chapter we describe a novel recommendation approach, developed based on the information collected from user-created lists, as a new way of discovering item-item relationships. We call this new approach, the Collective Intelligence ReCommendation (CIRC), since it draws upon the intelligence of the users creating cohesive collections as well as the opinion of the lists' viewers. CIRC operates on a preference model which comprises both user-item and item-item relationships, and infers a weighted item-item graph from the lists containing those items.

3.1 Our Model

Figure 3.1 illustrates the proposed list-based model for a book recommendation system. The model consists of three layers, the `User` layer, the `Item` layer and the `List` layer. We denote the set of `User` nodes by U, the set of `Item` nodes by I, and the set of `List` nodes by L .

A weighted edge connecting a `User` and an `Item` node is a triple (u, i, r_{ui}), where r_{ui} is the original rating in a 5-point scale range that u has given i. The set of all items a user u has already rated, is called the user's *basket*. An edge connecting an `Item` node and a `List` node is a pair (i, l) indicating that l contains i. The number of yes/no votes for each `List` node is assumed to be known. Therefore we have a set $V = \{(l_1, y_1, n_1), \ldots, (l_r, y_r, n_r)\}$, where a triple (l, y, n) says that y and n are the numbers of the "yes" and "no" votes, respectively, given to list l. List votes are anonymous, in other words, we do not know which users have voted for the lists, therefore there is no edge connecting user nodes to list nodes.

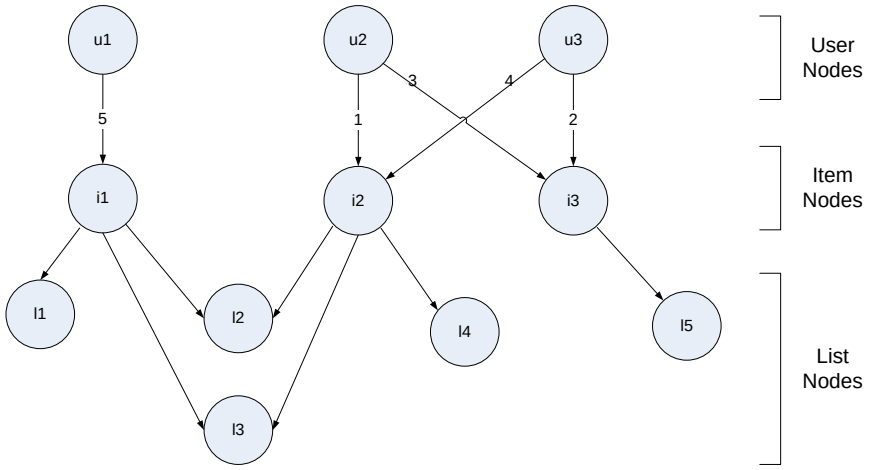

Figure 3.1: The proposed list-based model comprising user-item and item-item relationships

U, I, L along with (u, i, r_{ui}) and (i, l) form our knowledge base $\mathcal{K}$. Finally, we generate the set

$$\mathcal{A}_I = \{(a, b, w_{ab}) : a, b \in I \text{ and } w_{ab} \in \mathbb{R}^+\}$$

from item-list edges and list-vote triples.

The assumption here is that any two items which appear in a list together are *associated*. The weight w_{ab} is calculated in a way that reflects the strength of the association between items a and b. A weight of zero indicates that the two items are not associated, i.e., there is no list containing both items a and b. Set $\mathcal{A}_I$ represents an undirected weighted graph as shown in Figure 3.2.

Analyzing the co-occurrence of items is not currently a common method of analysis. [33] has suggested using the co-occurrence of artists in playlists shared by the users in a web-based music community in order to uncover the affinity of artists to multiple genres. Similarly in this research we rely on the co-occurrence of items in lists to infer the degree of association between items. We examine two different weights for the item pairs, namely, *frequency* and *sum of Bayesian ratings*. The latter takes into consideration not only the frequency of co-occurrence of a and b, but the number of votes that lists have received by anonymous users as well. We aim to show that it is not just the frequency of co-occurrence that is important but also the quality of the lists containing a pair of items. The precise formulas for calculating w_{ab} will be given in the next section.

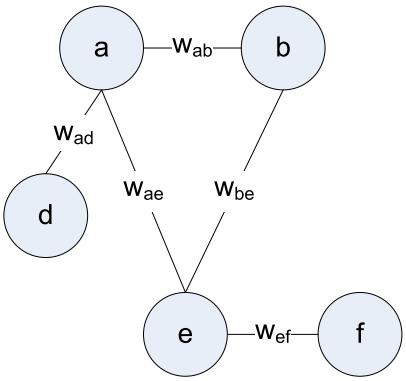

Figure 3.2: Item-item associations inferred from CIRC preference model

3.2 Weight Calculation

As we mentioned earlier, we examine two different weights for the item pairs, *frequency* and *sum of Bayesian ratings*. The rest of this section is devoted to the exact definitions as well as properties of these two weights.

Frequency. We define w_{ab} as the number of all lists in L containing both a and b.

The frequency of co-occurrence of items is a good indicator of possible associations between items, however, it's not just the frequency of co-occurrence that counts, but it's also the quality of co-occurrence. In other words, a pair of items which appears in a very desirable list is as significant as - and sometimes more important than- a pair appearing in many lists. Intuitively, if a user finds a useful and well put together list of items (e.g. a good playlist), he'd rather rate the list to express his interest than creating a new list containing the same items. This way, items that are really related will get more and higher votes and probably they will not appear together in many other lists.

We define Sum of Bayesian Ratings (SBR) as the second proposed weight describing the degree of association between two items. SBR takes into consideration not only the frequency of co-occurrence of the two items, but the number of votes that lists have received by anonymous users as well.

Sum of Bayesian Ratings (SBR). Sum of Bayesian ratings of all lists l in L containing both a and b:

$$w_{ab} = \sum_{(a,l) in \mathcal{K} \ and \ (b,l) in \mathcal{K}} BR(l)$$

Data Type	Number
C	the average rating of all the lists in knowledge base ‖
M	the average number of votes (including no votes) given to the lists in ‖
R_l	the rating of list l
N_l	the total number of votes given to list l

Table 3.1: Parameters used to calculate the Bayesian rating of a list

where

$$BR(l) = \frac{C \times M + R_l \times N_l}{M + N_l}$$

and C, M, R and N are defined in Table 3.1.

The rating of a list, R_l, is computed as the percentage of yes votes out of total votes given to l. The more desirable the list containing an item pair is, the more confident we are that these two items are related to each other. However, the desirability of lists can not be solely computed based on the percentage of yes votes they receive. Bayesian rating suggests that the number of votes given to a list is a key factor in determining the desirability of that list. If there are only few votes, then these votes should count less than when there are many votes. In other words, the more votes a list has, the higher the weight of these votes. Our experiments show that sum of Bayesian ratings used as weights on A, will result in producing better recommendations (see Chapter 5). The following example demonstrates how using Bayesian ratings changes the degree of association between items in two pairs with the same frequency.

Example. Consider two item pairs $p_1 = (a, b)$ and $p_2 = (c, d)$. Items in p_1 appear together in list l_1 with 1 "yes" votes (out of 1), and the items in p_2 appear together in list l_2 which has received 99 "yes" votes out of all its 100 votes. Since p_1 and p_2 appear in *one* list each, $w_{ab} = w_{cd} = 1$ if we use the frequency as weight. On the other hand, assuming $C = 0.46$ and $M = 9$, we will have $w_{ab} = 0.51 \neq w_{cd} = 0.94$ if we use SBR as weight. This example shows how considering the quality of the lists changes the degree of association between items.

3.3 Producing Recommendations

Given a target user, we produce recommendations based on the model we described in Section 3.1. The following algorithm computes the predicted rating of *selected* items

in I. Our recommendation for the target user is a subset of these items, i,e, top-N best rated items where N is the number of recommendations we need for a user.

Intuitively, CIRC algorithm starts with the items that the user has liked in the past (i.e. user's basket). Then it follows the links between these items and the items yet unseen by the user, predicts the ratings for the new items and finally chooses N best rated items as recommendations.

Input: A target user u, the desired number of recommendations N, minW, a knowledge base $\mathcal{K}$, and a set of item associations $\mathcal{A}_I$.

Output: A list of N item-prediction pairs (i, p_{ui}) for user u.

Method:

1. While $|\text{RESULT}| < N$ do:
   (a) For each item a in the u's basket do:
      i. Retrieve all neighbor items b of a in $\mathcal{A}_I$ such that $w_{ab} \geq minW$.
      ii. Let $neighbor(a)$ be the set of these neighbors.
      iii. For b in $neighbor(a)$ do:
         A. $nom(b) = nom(b) + r_{ua} * w_{ab}$
         B. $denom(b) = denom(b) + w_{ab}$
         C. Add j to the TEMP.
   (b) For each item b in the TEMP do:
      i. $p_{ub} = nom(b)/denom(b)$
      ii. Add (b, p_{ub}) to the RESULT
   (c) Let u's new basket be equal to TEMP
2. return RESULT.

CIRC algorithm computes the predicted rating of item i for user u as the following weighted average:

$$p_{ui} = \frac{\sum_{(i,j,w_{ij}) \in \mathcal{G'}_I} r_{uj} \cdot w_{ij}}{\sum_{(i,j,w_{ij}) \in \mathcal{G'}_I} w_{ij}}.$$

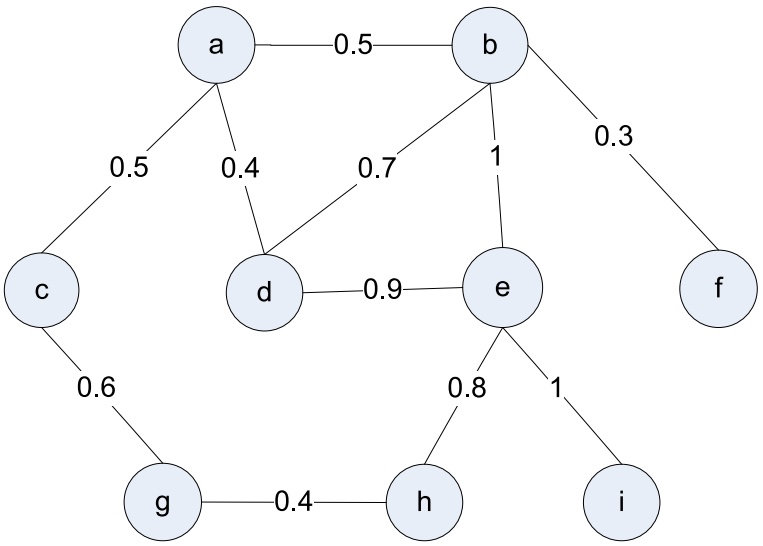

Figure 3.3: Item-item association graph $\mathcal{G}_I$

This is the same as the weighted sum defined in Equation 2.9 to predict ratings of unseen data utilized by item-based collaborative filtering algorithms. ($sim(p, p') = w_{pp'}$). However, CIRC recommendation is more *efficient* that item-based recommendation since CIRC does not compute the predicted rating for *all* the available items as is done in a basic item-based algorithm. In fact, CIRC predicts ratings for as many items as needed to find N items which the target user will like. When N items are found, CIRC stops predicting.

To illustrate how CIRC recommendation algorithm performs, we utilize the graph representation as shown in Figures 3.3 - 3.8. As it was mentioned earlier, we may represent the item association set $\mathcal{A}_I$ as a weighted undirected graph $\mathcal{G}_I$ (Figure 3.3). We extend graph $\mathcal{G}_I$ by adding a node representing the target user. This user node is connected to those items in the graph which are in user's basket (Figure 3.4). Initially the label of each item node in user's basket is the rating given to that item by the user and any other node has a 0 label.

CIRC aims at finding labels for the items in the graph which are not in the user's basket. We consider one item of user's basket at every step and update the labels of unseen items accordingly (Figures 3.5 and 3.6). Until all the items in user's basket are examined, each unseen item b keeps two temporary values $nom(b)$ and $denom(b)$ which are used to compute the actual label (predicted rating) for that items by the end of the first iteration of algorithm (Figure 3.7). If there are less labeled item nodes than the required number of recommendations, we iterate over the algorithm

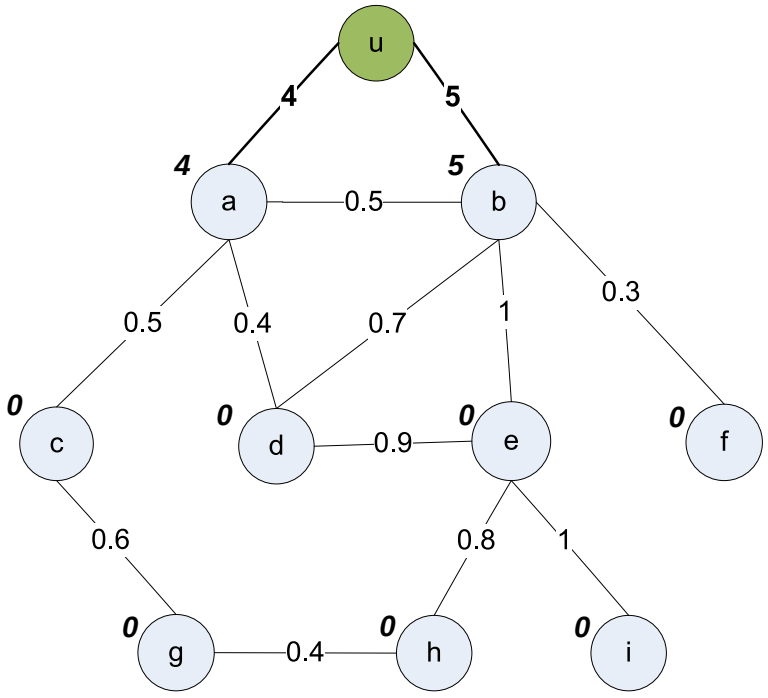

Figure 3.4: The extended graph $\mathcal{G}'_I$. User node u is connected to items in u's basket. Actual rating for each item in user's basket is shown in the upper left corner of the item node. Initial predicted ratings of other nodes are 0.

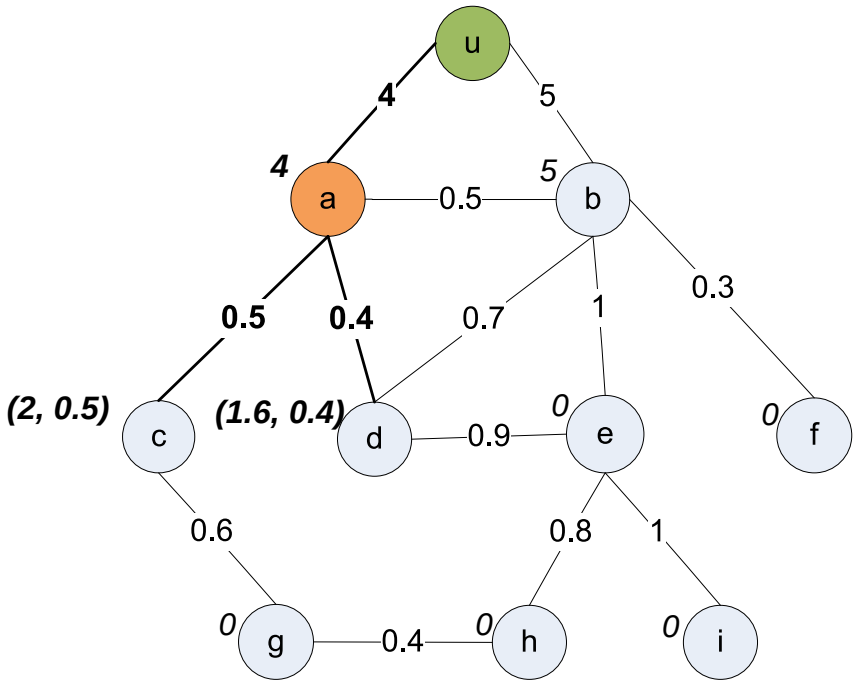

Figure 3.5: First iteration of algorithm over the first item in user's basket (a). Labels of items connected to a are updated. A pair of numbers shown outside items nodes represents $(nom, denom)$ temp values used by recommendation algorithm to predict the ratings after each iteration over user's basket.

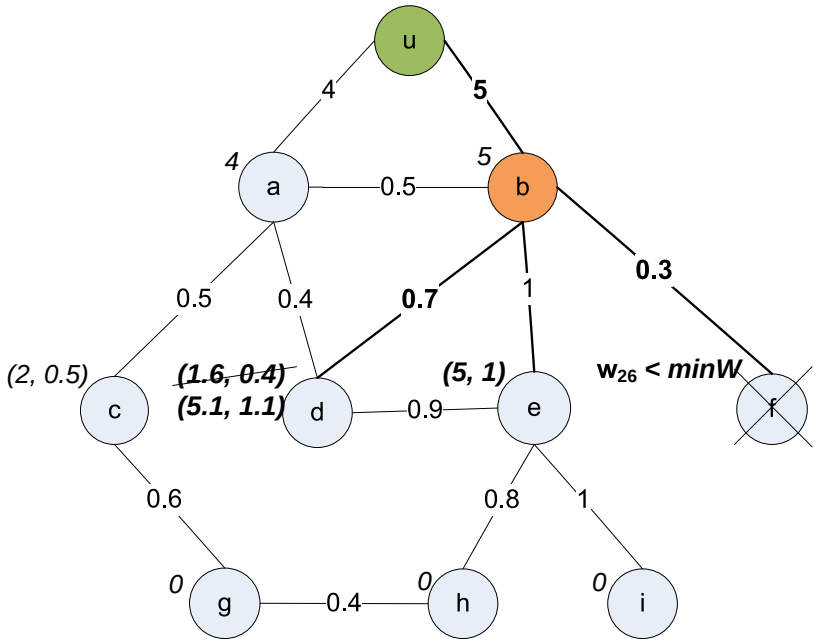

Figure 3.6: Second iteration of algorithm over the second item in user's basket (b). Label of item d is updated again because it is also in interaction with the second item in user's basket

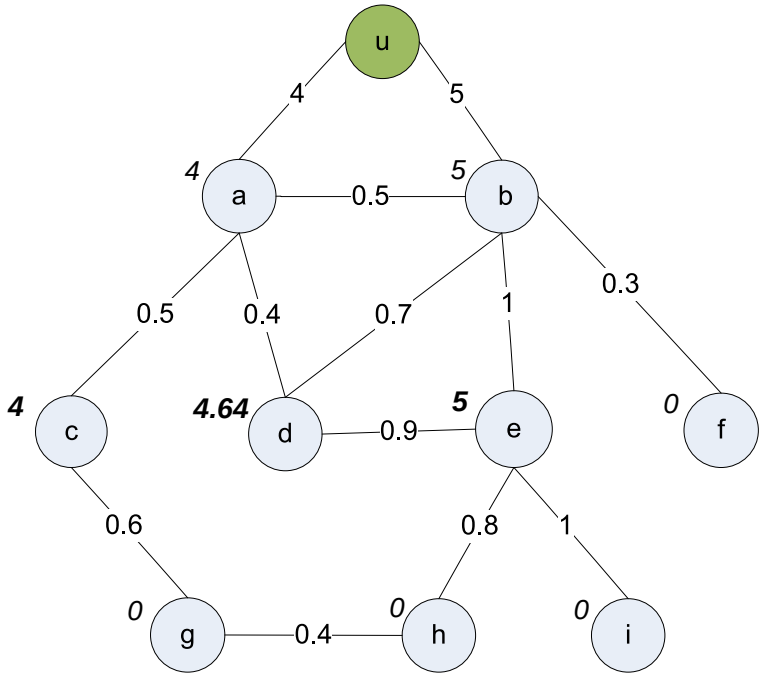

Figure 3.7: When all items in user's basket are traversed we calculate the predicted rating of unseen items based on information saved in their labels during the first and second iteration

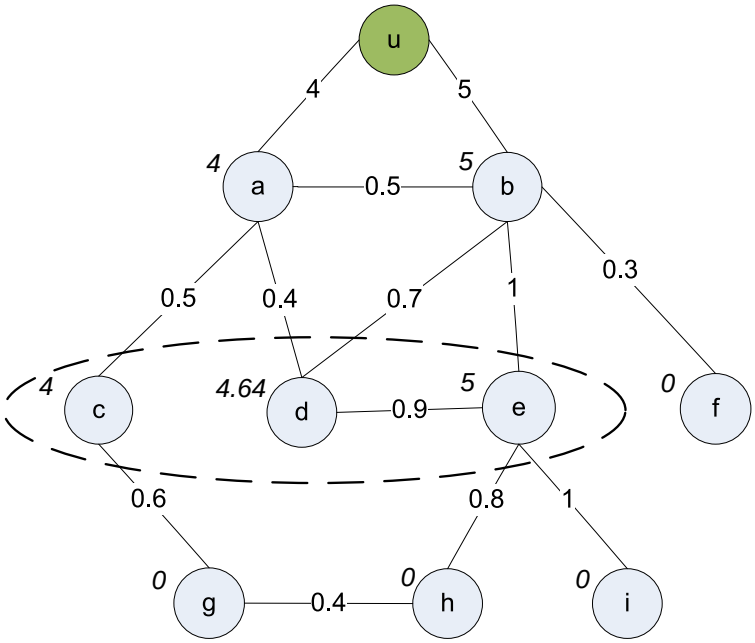

Figure 3.8: If the number of recommendations is not enough, we consider the set of newly rated items as user's new basket and resume the algorithm until N recommendations are found

by considering the set of newly labeled items as user basket (Figure 3.8).

In our algorithm we have a parameter $minW$ which denotes the minimum accepted value of w_{ab} for the items a and b to be considered associated. This threshold can be flexibly configured in CIRC for a specific application.

3.4 How to update CIRC?

Any changes in the knowledge base which is only associated with the set of items or the set of users, does not affect the item associations (i.e. $\mathcal{A}_I$). Examples of such changes are:

- When a user rates a new item or changes his opinion about a previously rated item
- When a new user is subscribed to the system
- When a new item is made available to users

CIRC needs to update item associations whenever a new list is added by a user or an existing list has been modified either by adding/deleting an item from the list or in case a new vote has been given to it. In all of these situations the updating procedure is fairly simple and may be done at negligible cost without requiring the system to stop its services:

- When a new item is *added* to a list, we should generate every possible pair of items from the items contained in the list making sure that at least one of the items in each pair is chosen from the set of newly added items. Some of these pairs are already in $\mathcal{K}$ for which we update the weights based on the new information now available. For the pairs that are new, we insert them in $\mathcal{K}$ and calculate the weights accordingly.

- *Deleting* some items from a list may cause a pair to completely disappear from $\mathcal{K}$. For any other remaining pair containing at least one of the deleted items, we need to update the weights associated with those pairs.

- Adding a *new list* has a similar effect on knowledge base $\mathcal{K}$ as adding new items to a list, except that we should generate every possible pair of items from the items contained in the new list without any constraint. (see Note below)

- When a *new vote* is given to one of the lists, all item pairs associated with that list will have new SBR weights based on the new vote. (See Note below)

Note: We should be aware that adding a new list to the system or giving a new vote to one of the lists, alters the constants defined by CIRC for calculating Bayesian ratings of the lists (see Table 3.1). More specifically, the average votes (C) and the average number of votes (M) calculated over all the list in the system are changed. These changes basically affect *every* SBR weight we calculated for item pairs. However, we may disregard these changes unless new averages are significantly different compared to the old averages. To define what is a *significant* difference, more research needs to be done on empirically analyzing CIRC when users actively interact with the system and receive recommendations.

Chapter 4

Evaluation

This section presents the experimental settings on validating the ability of CIRC to produce high quality recommendations.

4.1 Dataset

To evaluate the performance of CIRC we elaborated our novel dataset[1] using data gathered from Amazon.com[2] through Amazon Web Services (AWS)[3]. AWS is a web API which facilitates the task of retrieving publicly available information about users and items stored in Amazon servers.

We use a recursive process to gather the data. We start by first finding some random Listmania lists. Then we retrieve all the other Listmania lists containing the items found in the initial random lists. We continue recursively using these new lists until a considerable amount of Listmania lists is retrieved. Since we focus on book recommendations, we remove from the lists any non-book item. At this moment AWS does not provide yet access to list votes. Therefore, we conducted a crawling process to access list votes through HTML content.

Table 4.1 reports the type of data gathered and some statistics about this data. A pair (l, i), where l is a list and i is an item, indicates that i is in l. Note that ratings given in Table 4.1 are user preferences for items recorded in an integer 5-point scale and are different from list ratings (yes/no votes by anonymous people).

[1] http://webhome.csc.uvic.ca/∼maryamk/Amazon-Dataset.zip
[2] http://www.amazon.com
[3] http://www.aws.amazon.com

Data Type	Number
Item (Book)	405,238
User	530,160
List	58,618
User-Item-Rating	1,188,435
(l, i)	1,056,932

Table 4.1: Some statistics about the data set gathered from Amazon.com.

4.2 Method and evaluation metrics

We consider five sets (categories) of users based on the number of items they have rated. We randomly selected a 5% of the users in each subset as target users (or sample users). For each target user we use 80% of the items that he/she has rated as *input set* whereas the rest of items as *examination set*. The reason we chose the majority of items in user's basket as input set is the fact that it better corresponds to the deployment of the system in practice where all the ratings are used to produce recommendations and yet it reserves reasonable number of items in examination set to facilitate the evaluation process (see Apendix A.1). Table 4.2 reports statistics about the sample target users.

Category	No. of ratings R	Category population (P)	Sample size
C1	$R < 5$	497,664	24,884
C2	$5 \leq R < 10$	16,383	820
C3	$10 \leq R < 50$	9,834	492
C4	$50 \leq R < 500$	1,442	73
C5	$R \geq 500$	28	2

Table 4.2: Averaged Statistics about the sample target users

Since CIRC recommends to the user the items s/he might like, we only consider *positive ratings* [17] in the input set to produce recommendation. We call a rating on item p given by user u to be positive if $r_{u,p} \geq 4$. For users who had at least one positive rating in the input set and one in the examination set, we generated recommendations. A recommended item p to the user u is a successful recommendation if p appears in the examination set for user u and $r_{u,p} \geq 4$. To evaluate the performance of CIRC we use the common metrics used to evaluate recommendation algorithms [34, 35]:

- *Mean Absolute Error:* measures the average absolute difference between the

	Recommended=Yes	Recommended=No	Total
Liked=Yes	TP	FN	N_l
Liked=No	FP	TN	N_{nl}
Total	N_r	N_{nr}	N

Table 4.3: Confusion matrix for computing precision and recall

predicted rating of a recommended item and the true user rating for that item. If the user has rated N items, the MAE for him is calculated as:

$$MAE = \frac{\sum_{i=1}^{N} |p_i - r_i|}{N}$$

where p_i is calculated based on leave-one-out method.

- *Precision:* the portion of the recommended items the target user likes according to the examination set. To compute precision when users have binary preferences (like/dislike), a 2x2 table such as the one shown in Table 4.3 is used. The precision is computed as follows:

$$Precision = \frac{TP}{N_r}$$

where TP is the number of recommended item the target user actually liked and N_r is the total number of items recommended to the target user. We assume that the target user will consider all the items that are recommended.

- *Recall:* the portion of items the target user likes that are recommended to him/her. Recall actually presents the probability that a liked item will be selected and recommended to the user during the recommendation process. Referring to Table 4.3, Recall is computed as follows:

$$Recall = \frac{TP}{N_l}$$

- *F1:* combines precision and recall into a single metric by computing the harmonic mean of them. An interesting consequence of this definition is that F1 measure, tends to the least amount of precision and recall.

$$F_1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

- *Coverage:* measures the suitability of the recommender system in terms of the percentage of items for which the system is able to produce recommendations and predict ratings. To compute coverage we consider the items in which the target user may be interested, say, the items in target user's examination set. We then ask for a prediction for each item based on leave-one-out method and measure the percentage for which a prediction was provided.

We compute these five measures for every sample user. The final results reported for each category are then averages over all users of that category.

Chapter 5

Evaluation, Analysis and Comparisons

This section discusses experimental results on recommendations produced by CIRC. For comparison purposes, we have also included the results of both item-based and user-based collaborative filtering recommendation systems which employ the Pearson nearest neighbor algorithm. For these two collaborative filtering algorithms we use 80% of the data set as training set and 20% as test set to be consistent with the parameter n we fixed for CIRC.

In what follows we first detail the result of two experiments. The first experiment aims at showing that the frequency of co-occurrence of items is a good indicator of possible associations between items but it is not necessarily enough to just rely on frequency (Section 5.1).The second experiment evaluates the recommendation quality produced by CIRC using *sum of Bayesian ratings* as weight, and reports the improvements achieved compared to the results of the first experiment (Section 5.2). Finally in Section 5.3 we compare CIRC to previously mentioned user-based and item-based approaches using the metrics introduced in section 4.2.

5.1 Frequency as weight

This experiment employs pair frequencies as weights and explores various frequencies as the value for the parameter $minW$. As it was mentioned before, CIRC has a parameter $minW$ which can be flexibly configured. Items with higher association values (i.e. w_{ab}) than $minW$ are considered to be neighbors and therefore contribute

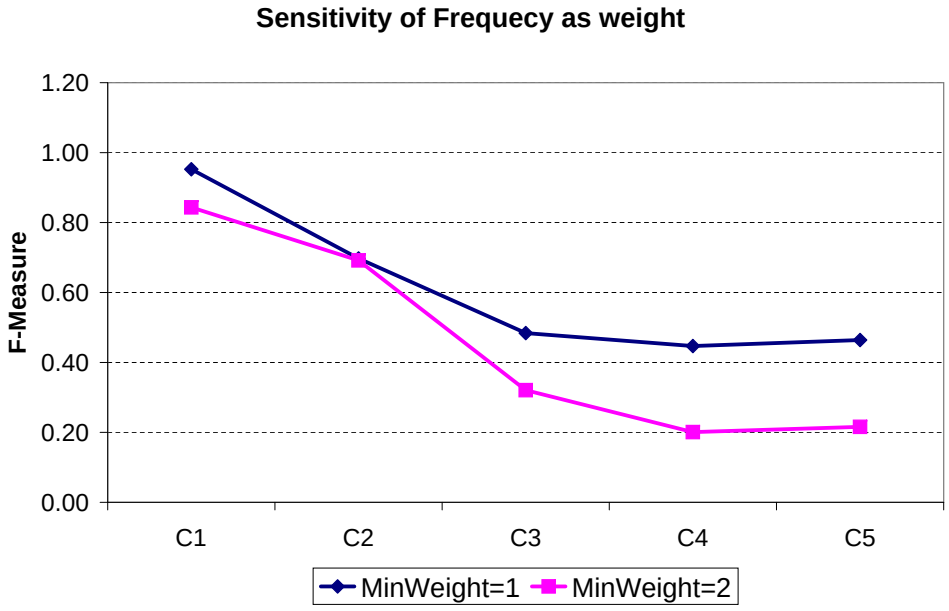

Figure 5.1: Recommendation quality of CIRC using frequency as weights

in the process of recommendation, otherwise, the pair is believed to be not related.

Figure 5.1 shows the quality of recommendations produced by CIRC for different values of $minW$ when frequency is used as weight. We can observe the effect of varying the value of $minW$ (i.e. $minFrequency$) on recommendation quality. By increasing the threshold from one to two, the quality of recommendations drops significantly especially in categories `C3`, `C4` and `C5` where the users have rated more items. One explanation could be that by ignoring the pairs of items who have just appeared once together in one of the lists, we are actually missing valuable information.

This observation supports the idea that it's not just the frequency of co-occurrence that counts, but it's also the quality of co-occurrence. In other words, a pair of items which appears in a very desirable list with high percentage of yes votes and high number of voters, is as significant as - and sometimes more important than- a pair appearing in many lists. Intuitively, if a user finds a useful and well put together list of items (e.g. a good playlist), he'd rather rate the list than creating a new list containing the same items to express his interest. This way, items that are really related will get more and higher votes and probably they will not appear together in many other lists.

When using frequency as weight, one may want to configure CIRC with higher thresholds of $minW$, firstly because low frequency is usually an indicator of low associations between items in a pair, and secondly, to reduce the response time of the

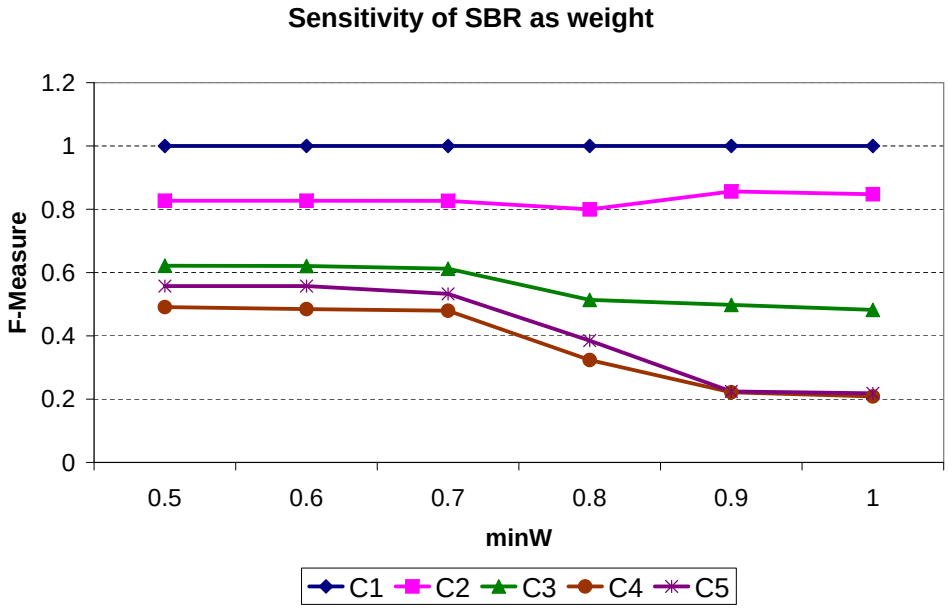

Figure 5.2: Recommendation quality of CIRC using sum of Bayesian ratings as weights

recommendation algorithm by filtering uninteresting pairs out. However, it is possible for an item pair which consist of highly related items to have a low frequency. Setting a high threshold also filters these good pairs out. Therefore, there should be a way to distinguish between good pairs and bad pairs with the same frequency without compromising the quality of recommendations or exceeding the optimal response time. The following section discusses the alternative weight (i.e. sum of Bayesian ratings) for dealing with the aforementioned problems.

5.2 Sum of Bayesian ratings (SBR) as weight

In this experiment we use a more sophisticated approach to decide between an interesting pair and the ones who imply low association between items. Based on the discussions on Section 3.2, we employ the sum of Bayesian ratings as the weight to describe a pair of items.

Figure 5.2 illustrates the quality of recommendations produced by CIRC using sum of Bayesian ratings as weight. It can be observed from the results that increasing the $minW$ has negligible effect on the quality of recommendations for the first three categories. While categories `C4` and `C5` show little change in quality when $minW$ increasing from 0.5 up to 0.8, they are more affected by further increase in $minW$

compared to the first three categories. We believe that the low number of sample users in the last two categories contributes to the sensitivity of the results, however, further research should be conducted to study this behavior in more detail.

This observation helps us to tune our recommender system in a way that generates the best possible recommendations. In general, unlike using frequency as weight, where the quality sharply drops by increasing the $minW$, interesting pairs are preserved by using SBR even when the threshold is high. This means that SBR is a better indicator of existing associations between items and better separates the interesting items from the items the user might not like. Two pairs with the same frequency may have different association degrees due to the quality of lists they have appeared in.

By setting higher thresholds for $minW$ parameter, there remain fewer pairs satisfying the new threshold. Therefore, we look into a smaller set of pairs to find recommendations for a user and this reduces the time needed for producing recommendations while keeping the high quality. This is an interesting and promising observation that is of great importance to real-time recommender applications.

This experiment justifies the use of the novel SBR weight for better anticipating the associations between items. We further compare the performance of CIRC using SBR with that of CIRC using frequency as well as user-base and item-based recommender algorithms in the following subsection. For this matter, we will set $minW$ equal to 0.5 to get the best possible results CIRC produces while using SBR.

5.3 CIRC vs. item-based and user-based collaborative filtering

We categorize CIRC as a model-based recommendation approach which utilizes the information provided by users in a new way to derive the item-item relationships. While it seems similar to an item-based collaborative approach, it's genuinely different because CIRC does not consider the user-item interactions in users' profiles. We believe that in applications where extra data like user-created lists is available, CIRC can outperform commonly used user-based and item-based approaches.

Figures 5.3 - 5.5 report the results of comparing the performance of CIRC with that of user-based and item-based recommendation approaches. We include the results of CIRC using two different weights we introduced before so that we can compare two

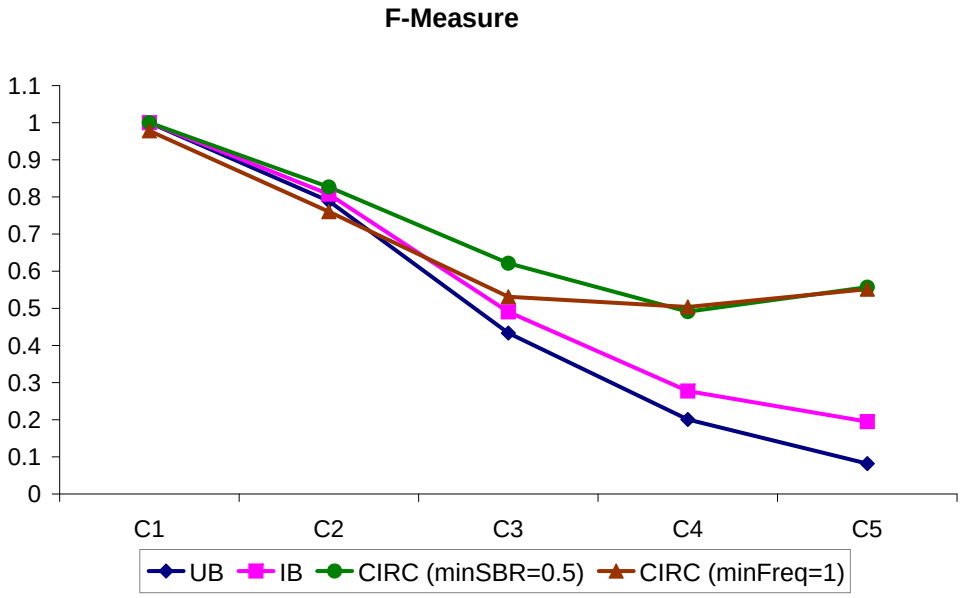

Figure 5.3: Comparison of classification accuracy of CIRC and user-based and item-based algorithms

versions of CIRC with respect to each other as well as with user-based and item-based approaches. We report F-measure as an indicator of system accuracy in finding items the user will like, while MAE can be interpreted as the confidence an algorithm has in recommendations it makes and coverage measures the usability of the system to the users by measuring the percentage of the the items in a user's basket for which a prediction was provided. This computation is based on the leave-one-out method. Precision and Recall measures which are not shown in the figures, are reported in Appendix A.2.

F-measure. It can be observed from the charts in Figure 5.3 that at least one of the two *CICR* algorithms (frequency, SBR or both) provides more accurate recommendations according to the reported F-measures. The distinction between our approach and two benchmark CF algorithms that we have implemented becomes even more apparent for categories `C3`, `C4` and `C5` where users have rated more items. We believe this happens partly because as the users rate more and more number of items there is a risk of mixed tastes in their generated neighborhood. Therefore, lots of recommended items are of no or lower interest to the user. CIRC on the other hand, remains on track by just looking at quality item-item associations and is not confused by a misleading neighborhood.

CIRC better finds good items and filters out uninteresting items when it uses SBR

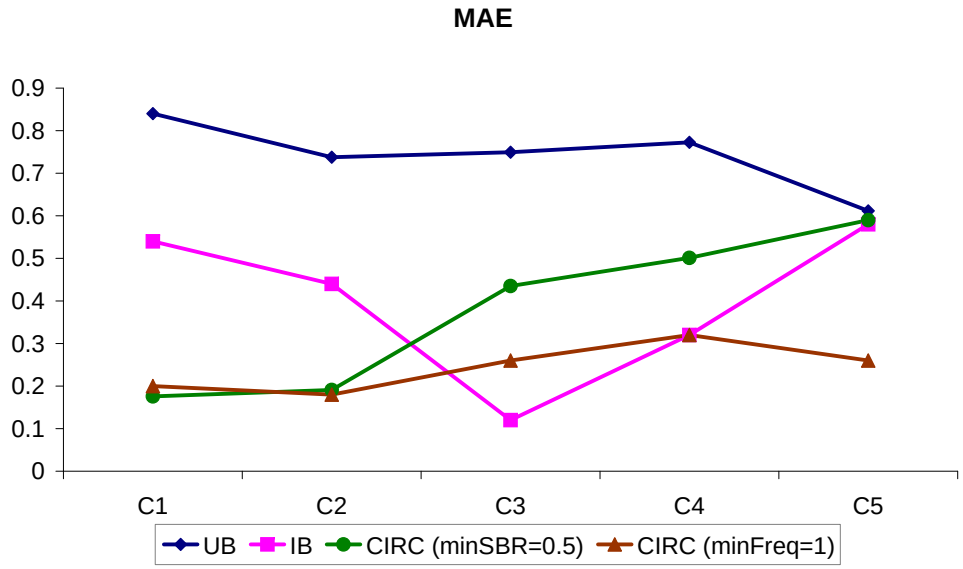

Figure 5.4: Comparison of prediction accuracy of CIRC and user-based and item-based algorithms

as opposed to frequency. Therefore we draw this conclusion that, collections (or lists) of items created by users along with the opinion of voters about these collections, is an important and promising information source to derive the meaningful associations between items.

MAE. Figure 5.4 reports the quality of predictions made by each of four algorithms under examination. While user-based algorithm always bears the worst prediction quality of all four, item-based approach shows an interesting behavior and even outperforms CIRC predictions in category `C3`. Starting from category `C1` with lowest number of rated items for each user, the quality of predictions increases as the sparsity of data set decreases. However, this increase does not last and it stops at `C3` which we believe is the best case dataset for item-based algorithm. Afterwards we observe the quality going down. Except for the third category of users, CIRC is the winner in predicting the ratings for recommended items. However, unlike classification accuracy, prediction accuracy is higher when we use frequency as weight. This is an interesting observation which can be addressed in more details in future research.

Coverage. While F-measure and MAE are important, we are also interested to find out how useful a system is to the users. In the other words, what percentage of users will receive recommendations if any. Figure 5.5 illustrates the usability of these four recommender algorithms in terms of the coverage they provide for different user categories. Without any exception CIRC outperforms item-based and user-based ap-

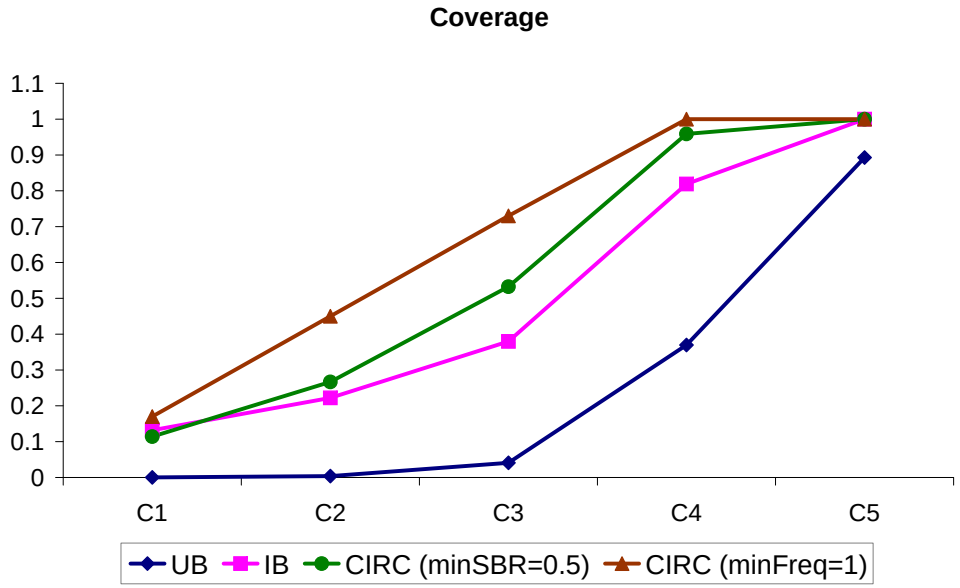

Figure 5.5: Comparison of coverage of CIRC and user-based and item-based algorithms

proaches. We also observe that frequency provides more coverage than SBR. This is not surprising since we have set the $minFrequency = 1$ which means all the possible item-item relationships will be examined, increasing the chance to produce recommendations for more users. For the SBR however, we chose $minSBR = 0.5$ because this is the value which yields highly accurate recommendations without requiring the system to look into enormous amount of data. We know that by setting lower thresholds for $minSBR$ we will get the same coverage as what we get using frequency as weight.

Discussion. From the experimental evaluations of CIRC along with two benchmark algorithms, user-based and items-based collaborative filtering schemes, we make some important observations. First, as it was expected the item-based technique always outperforms the user-based approach by having higher F measure, higher coverage and lower error for predicting ratings. The second observation is that CIRC is almost always the winner except for the yielded prediction accuracy of category `C3` in which item-based recommender performs better. It is important to note that all measures used to evaluate this work suffer from the underlying biases as is suggested by Herlocker et al [34]. However, to alleviate this bias in our evaluations, we only used the portion of the recommendations which also appear in the target user's examination set, otherwise, a recommender with high *true recall* and *true precision* may yield low values on these measures because it recommends lots of *un-rated* relevant items

to the user. The comparisons, however, are absolute and show the prominence of our approach over benchmark algorithms. By removing this constraint in real world application of CIRC we expect to have even more coverage.

We implemented all our experiments using JAVA on a Windows XP based PC with Intel Dual core processor having a speed of 2GHz and 2GB of RAM. Although we have not done any experiment on scalability issues like response time and throughput, we are confident that CIRC is highly scalable due to the way it produces recommendation. As discussed earlier all the item-item relationships are pre-computed and kept up to date at a negligible cost (see section 3.4).

Chapter 6

Conclusions

6.1 Summary and Conclusion

In this thesis we proposed a new model for recommendation systems based on valuable information present in user-created lists which is usually overlooked. We described how it is possible to extract item-item associations from these lists and use these associations to make recommendations in a more efficient and qualitative way. Our assumption is that the co-occurrence of two items in a list together is a strong indicator of the similarity between those items. This assumption is intuitively supported by the fact that user-created lists (e.g play lists, book collections, movie collections, etc.) usually have a unique theme, taste, and topic. To evaluate this assumption we introduced two ways to calculate the degree of association between two item: *frequency of co-occurrence* and *sum of Bayesian ratings*. The latter takes into consideration not only the frequency of co-occurrence but also the quality of the lists containing the item pair. We experimentally evaluated CIRC- our Collective Intelligence ReCommendation approach based on the proposed model- and showed that CIRC outperforms the commonly used user-based and items-based CF methods. Moreover, the results suggest that when "sum of Bayesian ratings" is used as item-item similarity measure, CIRC produces more accurate recommendations, covers a broader range of users and items and has higher confidence in what it recommends. CIRC is flexible and can be kept up to date at negligible costs. CIRC also deals with one of the most challenging problems with recommenders: the *cold start* problem; It performs remarkable well for cold-start users in our data set. All these results are very promising and suggest that this research has successfully discovered a new potential to extract item-item

associations. Our novel data set used in this research is publicly available.

6.2 Future Work

In this thesis our primary goal was to show that user-created lists of items contain valuable information for extracting item-item relationship. As a secondary goal we wanted to show that the co-occurrence of two items in lists is not the only important issue, but it is also the quality of the list two items appear in together. We calculated the "Bayesian rating" of the lists as a quality measure. However, we could simply define the quality of a list as the percentage of yes votes out of all the votes it has received. It will be interesting to study the behavior of CIRC using this quality measure and to compare the results with that of CIRC when frequency and sum of Bayesian ratings are used as measures. If the former shows superior performance, then updating CIRC is even less expensive. The reason is that in this case the quality of lists never depend on a global constant like C (average rating of all the lists) and M (average number of votes received by lists). Therefore, we never have to re-calculate any item-item similarity. Also, our research does not include any performance analysis in terms of the response time and system throughput.

We have shown in this research that when extra information about user-created lists is available, systems can benefit from CIRC. However, CIRC can also be utilized in conjunction with other recommendation methods to form a hybrid approach.

Appendix A

Additional Information

A.1 Parameters

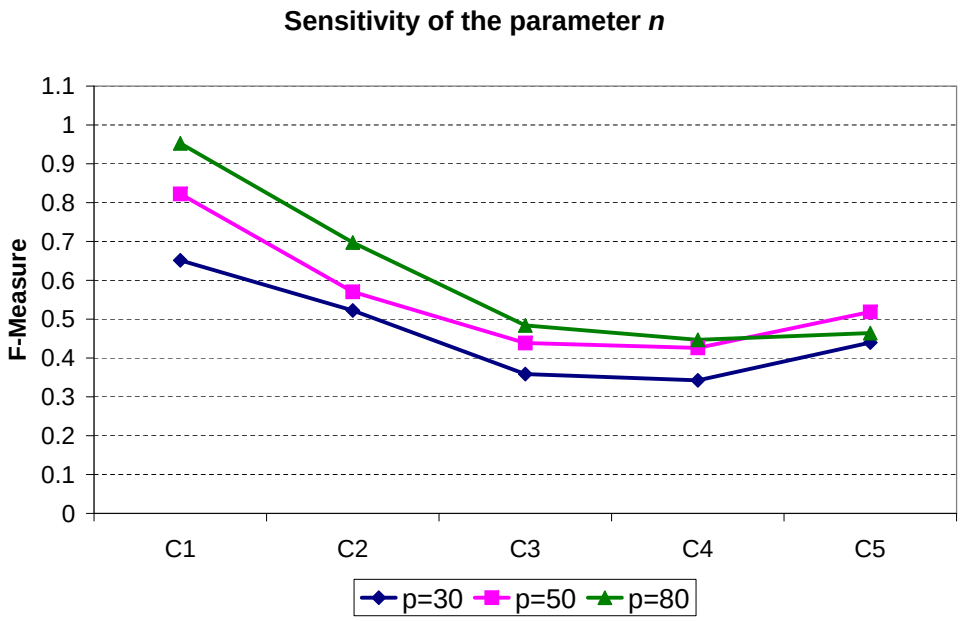

Figure A.1: Sensitivity of the parameter n

As it was mentioned in section 4, in our experiments with CIRC we use the majority of items in user's basket as input set. We carried on a preliminary experiment where we used three different values (30, 50, 80) as the value of n. In our experiment n indicates the percentage of items in the user's basket to be used as input set while the rest of the items serve as the examination set. To determine the sensitivity of parameter n, we ran CIRC for each of these values using frequency as weight and

fixing MinW=1 and we computed the F-measure. Our results are shown in Figure A.1. We observe that the quality of recommendations increases as we increase n. Based on this, we select n=80% as an optimum value for our subsequent experiments with CIRC.

A.2 More Results

The following table reports the precision and recall resulted from experiments.

Category	User Based		Item Based		CIRC minFreq=1		CIRC minSBR=0.5	
	P	R	P	R	P	R	P	R
C1	1.00	1.00	1.00	1.00	1.00	0.96	1.00	1.00
C2	0.88	0.72	0.91	0.73	0.91	0.65	0.92	0.75
C3	0.76	0.30	0.78	0.36	0.82	0.39	0.88	0.48
C4	0.76	0.12	0.75	0.17	0.85	0.36	0.82	0.35
C5	0.87	0.04	0.86	0.11	0.71	0.45	0.73	0.45

Table A.1: Precision and Recall for four different recommendation algorithms examined

Bibliography

[1] O. Celma and P. Lamere, "Music recommendation tutorial," in *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR)*, September 2007.

[2] 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*, pp. 210–217, 1995.

[3] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," *Commun. ACM*, vol. 35, no. 12, pp. 61–70, 1992.

[4] 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*, pp. 175–186, 1994.

[5] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herloker, L. R. Gordon, and J. Riedl, "Grouplens: applying collaborative filtering to usenet news," *Communications of the ACM*, vol. 40, no. 3, pp. 77–87, 1997.

[6] M. Deshpande and G. Karypis, "Item-based top-n recommendation algorithms," *ACM trans. info. syst.*, vol. 22, no. 1, p. 143, 2004.

[7] 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*, pp. 285–295, 2001.

[8] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Application of dimensionality reduction in recommender systems-a case study," in *ACM WebKDD 2000 Web Mining for E-Commerce Workshop*, 2000.

[9] S. Deerwster, S. T.Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society for Information Science*, vol. 41, no. 6, p. 391, 1990.

[10] T. Hofman, "Latent semantic models for collaborative filtering," *ACM trans. info. syst.*, vol. 22, no. 1, p. 89, 2004.

[11] B. J. Mirza, B. J. Keller, and N. Ramakrishnan, "Studying recommendation algorithms by graph analysis," *J. Intell. Inf. Syst.*, vol. 20, pp. 131–160, March 2003.

[12] Z. Huang, H. Chen, and D. Zeng, "Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering," *ACM trans. info. syst.*, vol. 22, no. 1, p. 116, 2004.

[13] Z. Huang, X. Li, and H. Chen, "Link prediction approach to collaborative filtering," in *JCDL '05: Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries*, pp. 141–142, 2005.

[14] C. Basu, H. Hirsh, and W. Cohen, "Recommendation as classification: Using social and content-based information in recommendation," in *In Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pp. 714–720, 1998.

[15] L. H. Ungar and D. P. Foster, "Clustering methods for collaborative filtering," in *Workshop on Recommender Systems at 15th National Conference on Artificial Intelligence*, 1998.

[16] W. Lin, S. A. Alvarez, and C. Ruiz, "Efficient adaptive-support association rule mining for recommender systems," *Data Min. Knowl. Discov.*, vol. 6, pp. 83–105, January 2002.

[17] C. W. ki Leung, S. C. fai Chan, and F. lai Chung, "An empirical study of a cross-level association rule mining approach to cold-start recommendations," *Know.-Based Syst.*, vol. 21, pp. 515–529, October 2008.

[18] J. Golbeck, "Tutorial on using social trust for recommender systems," in *RecSys '09: Proceedings of the third ACM conference on Recommender systems*, (New York, NY, USA), pp. 425–426, ACM, 2009.

[19] M. Khezrzadeh, A. Thomo, and W. W. Wadge, "Harnessing the power of favorites lists for recommendation systems," in *proceedings of the 3rd ACM Conference on Recommender Systems*, (New York City, NY, USA), October 2009.

[20] G. Adomavicius and E. 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*, vol. 17, pp. 734–749, 2005.

[21] N. Littlestone and M. K. Warmuth, "The weighted majority algorithm," 1992.

[22] M. Pazzani, D. Billsus, S. Michalski, and J. Wnek, "Learning and revising user profiles: The identification of interesting web sites," in *Machine Learning*, pp. 313–331, 1997.

[23] W. Hill, L. Stead, M. Rosenstein, and G. Furnas, "Recommending and evaluating choices in a virtual community of use," in *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, (New York, NY, USA), pp. 194–201, ACM Press/Addison-Wesley Publishing Co., 1995.

[24] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pp. 43–52, Morgan Kaufmann, 1998.

[25] T. Hofmann, "Collaborative filtering via gaussian probabilistic latent semantic analysis," 2003.

[26] M. Balabanovic and Y. Shoham, "Fab: Content-based, collaborative recommendation," *Communications of the ACM*, vol. 40, pp. 66–72, 1997.

[27] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin, "Combining content-based and collaborative filters in an online newspaper," in *In Proceedings of ACM SIGIR Workshop on Recommender Systems*, 1999.

[28] M. J. Pazzani, "A framework for collaborative, content-based and demographic filtering," *ARTIFICIAL INTELLIGENCE REVIEW*, vol. 13, pp. 393–408, 1999.

[29] A. I. Schein, A. Popescul, R. Popescul, L. H. Ungar, and D. M. Pennock, "Methods and metrics for cold-start recommendations," 2002.

[30] I. S. C. Nicholas and C. K. Nicholas, "Combining content and collaboration in text filtering," in *In Proceedings of the IJCAI99 Workshop on Machine Learning for Information Filtering*, pp. 86–91, 1999.

[31] D. Billsus and M. J. Pazzani, "User modeling for adaptive news access," 2000.

[32] T. Tran and R. Cohen, "Hybrid recommender systems forelectronic commerce," in *In Proceedings of Knowledge-Based Electronic Markets, Papers from the AAAI Workshop*, (Menlo Park, CA), pp. 78–83, AAAI Press, 2000. AAAI Technical Report WS-00-04.

[33] C. Baccigalupo, J. Donaldson, and E. Plaza, "Uncovering affinity of artists to multiple genres from social behaviour data," in *International Conference on Music Information Retrieval* (D. T. Juan Pablo Bello, Elaine Chew, ed.), (Philadelphia, Pennsylvania USA), Drexel University, Drexel University, 14/09/2008 2008.

[34] J. L. Herlocker, L. G. T. Joseph A. Konstan, and J. T. Ridel, "Evaluating collaborative filtering recommender systems," *ACM trans. info. syst.*, vol. 22, no. 1, p. 5, 2004.

[35] Z. Huang, D. Zeng, and H. Chen, "A comparative study of recommendation algorithms in e-commerce applications," *IEEE Intelligent Systems*, vol. 22, no. 5, pp. 68–78, 2007.