A SCALABLE COLLABORATIVE FILTERING SYSTEM USING DISTANCE MEASURES OF SOCIAL NETWORK

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

İSMAIL MELIH ÖNEM

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

OCTOBER 2013

Approval of the thesis:

A SCALABLE COLLABORATIVE FILTERING SYSTEM USING DISTANCE MEASURES OF SOCIAL NETWORK

submitted by **İSMAIL MELIH ÖNEM** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Adnan Yazıcı
Head of Department, **Computer Engineering** _____

Prof. Dr. Ferda Nur Alpaslan
Supervisor, **Computer Engineering Department, METU** _____

Examining Committee Members:

Prof. Dr. Mehmet Reşit Tolun
Computer Engineering Department, TEDU _____

Prof. Dr. Ferda Nur Alpaslan
Computer Engineering Department, METU _____

Assoc. Prof. Dr. Çiğdem Turhan
Software Engineering Department, Atılım University _____

Assoc. Prof. Dr. Ahmet Coşar
Computer Engineering Department, METU _____

Dr. Ayşenur Birtürk
Computer Engineering Department, METU _____

Date: **03.10.2013**

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: İSMAIL MELIH ÖNEM

Signature :

ABSTRACT

A SCALABLE COLLABORATIVE FILTERING SYSTEM USING DISTANCE MEASURES OF SOCIAL NETWORK

Önem, İsmail Melih

M.S., Department of Computer Engineering

Supervisor : Prof. Dr. Ferda Nur Alpaslan

October 2013, 96 pages

Recommender systems are very popular in information systems and in the research community, where many different approaches geared towards giving better recommendations have been proposed. In this thesis, we propose a methodology that uses social network information to improve the performance of recommender systems. Our proposed methodology heuristically improves the success rate and performance of recommendation algorithms using social distance measures on a dataset that comprises people in professional occupations. Further, we explain how these methods apply to on-line real-world applications. The main objective behind the composition is to provide better and more relevant inputs to item-to-item filtering algorithms. We propose a compound method comprising three steps. In the first step, the algorithm elaborates social network distances and friendships to help recommender systems customize the target user set. To find people who are similar to a specific user, the system divides each worker's friends (target set) into subsets and treats the task as a social clustering problem. In the second step, clustering is done on social measures. The clustering algorithm divides the target set into subsets to build a job-to-job table and a similar-job pairs of people who tend to do the same kind of work. In the third step, highly recommended jobs are defined by computing distance metrics on job vectors. Thereby, item-to-item recommendation can compute ordered predictions for users. We interpret the differences between social-based relations and the impact of similarity metrics on a collaborative recommendation algorithm.

The experiments conducted on large datasets indicate that our proposed approach, which customizes recommendations using social connections, outperforms generic methods in terms of specificity and scalability. We also conducted several experiments to compare the evaluation and recommendation qualities of our approaches with other well-known algorithms such as Restricted Boltzmann Machines. Our evaluations show that the components of our method combine to facilitate deeper understanding of the performance characteristics of recommender systems.

Keywords: Recommendation System, Collaborative Filtering,Social Network, Hybrid Collaboration, Item to Item Recommendation, Boltzmann Machines

ÖZ

SOSYAL AĞ UZAKLIKLARINI KULLANILDIĞI ÖLÇEKLENDİRİLEBİLİR KOLLEKTİF FİLTRELEME SİSTEMİ

Önem, İsmail Melih

Yüksek Lisans, Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi : Prof. Dr. Ferda Nur Alpaslan

Ekim 2013 , 96 sayfa

Tavsiye Sistemleri bilgi sistemlerinde ve araştırma topluluklarında birçok farklı yaklaşımla daha iyi sonuç vermeyi hedeflemeleri ile çok popülerlerdir. Bu tez içerisinde, sosyal ağ bilgilerini kullanarak öneri sistemleri geliştirmek için bir metodoloji sunulmaktadır. Kişilerin iş bilgilerinin yer aldığı bir veri kümesi içerisinde denemeler yaparak, tavsiye algoritmasının başarı oranını ve performansını arttırmayı amaçlarken, bunu da gerçek dünyada nasıl uygulanabileceği gösterilmektedir.Hedef olarak da iş arayan kişiler için uygun iş fırsatlarını bulmayı sağlayan gerçek zamanlı bir sistemin oluşturulmasının mümkün olup olmayacağı araştırılmaktadır. Kompozisyonun arka planında filtreleme algoritmasına daha uygun ve anlamlı girdi vermek yatmaktadır.Algoritmanın ilk aşamasında, sosyal ağ uzaklıklarının ve arkadaşlık mesafelerinin hedef kümeyi özelleştirerek tavsiye sistemine nasıl daha uygun hale getirilebileceği araştırılmaktadır. Kullanıcıya benzer diğer insanları bulmak için sistem diğer insanları küçük kümelere bölmekte ve buna bir kümeleme(clustering) problemi olarak yaklaşmaktadır.Aslen ikinci aşamada bu kümeleme sosyal ağ uzaklıkları üzerinde yapılmaktadır. Kümele algoritması hedef kümeyi ilgili küçük kümelere bölerek, genel bir iş matrisi ve kişinin çalışma ihtimali olan iş çiftlerini oluşturmaktadır. Son olarak da, en çok tavsiye edilen işler, matris üzerinde yer alan vektörlerin birbirine uzaklıkları hesaplanarak çıkarılmaktadır. Item to Item Tavsiye Algoritması son adımda derecelendirilmiş bir tavsiye de yapmaktadır. Aslında, sosyal tabanlı ilişkilerin ve vektör benzerlik ölçütlerin algoritma üzerinde etkisi yorumlanmaktadır.

Ayrıca değerlendirmeyi ve tavsiye kalitemizi karşılaştırmak için Kısıtlanmış Boltzmann Makinası(RBM) algoritması da kullanılmaktadır. Geniş veri kümelerinde hedef kümeyi küçültmek performansı arttırmanın yolu olduğu için, genel yöntemlere göre ölçeklendirilebilir bir yaklaşım sunulmaktadır. Aldığımız sonuçlar, oluşturduğumuz kollektif filtreleme algoritmasının performans karakteristiği ve başarımı üzerinde derin bir kavrama yaratmıştır.

Anahtar Kelimeler: Tavsiye Sistemleri, Kollektif Filtreleme, Sosyal Ağlar, Hibrid Kollektif Filtreleme, Nesne-Nesne Tavsiye Algoritması, Boltzmann Makinası

To Loyal Friends

ACKNOWLEDGMENTS

I would like to thank to my supervisor Prof. Dr. Ferda Nur Alpaslan for her guidance,advice and criticism throughout the research. I also want to thank to the other committee members for their comments and suggestions.

I would like to express my deepest gratitude to B. Kozluca, who encouraged and supported me in this study.

I would also like to thank to colleague for donating a strong server computer to support my research.

Thanks are also to all of my friends for their direct or indirect help.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

CF	Collaborative Filtering
RBM	Restricted Boltzmann Machines
CD	Contrastive Divergence
FD	Friend Distance Measure
MF	Mutual Friend Measure
FMF	Friend's Mutual Friends Measure
WF	Weighted Friends Measure
COS	Cosine Distance
EUC	Euclidean Distance
MANH	Manhattan Distance
MINS	Minkowski Distance
ROC	Receiver Operating Characteristic (ROC), or Simply ROC Curve
AUC	Area Under the Curve
RMSE	Root Mean Square Error

CHAPTER 1

INTRODUCTION

1.1 RESEARCH MOTIVATION

Recommendation techniques generally give the consumer a list of recommended items he/she can possibly choose, or figure out how much he/she can choose each item. The objective of this research is to examine and improve the success rate of recommendation algorithms by using social distance measures on a dataset that comprises people in professional occupations, and also to explain how these methods apply to on-line real-world applications. The dataset used is a specific database comprising the social network of business people in Turkey. It stores the professional profile of each user, lists their present and past employment, and links them to former and current colleagues and classmates. It also has detailed information about companies and their properties. Information about present and past employment helps the recommendation algorithm to evaluate the results appropriately.

All existing recommendation methods suggest products or services to people. Whereas method presented in this thesis recommends one worker to another co-worker. The whole idea of applying recommender systems in social network is innovative, thus many new challenges have occurred. Building such a framework requires integrating two separate branches of knowledge: computer science and sociology. Moreover, when we recommend one worker to another, we should have deep knowledge about algorithms that can be utilized and pick the appropriate one. Generally, the chosen method must be further tailored to the specific needs of the system.

In this work, we focus specifically on the collaborative filtering problem and look at ways in which the increasing amount of social network data available in the current web can be utilized for this purpose. We believe that social networks will gain even more importance for use in information filtering applications. The main aim of this thesis is to create the recommendation method that will support the evolution of professional social networks and the objectives of the thesis are as follows:

- Gain knowledge about the subject (recommender systems and social networks) through research in literature

- Create the classification and clustering of existing social networks
- Build the employment profile of user in which different type of data will be crawled
- Define the recommendation process for social network that concerns the preferences and connections who will be recommended

1.2 METHODOLOGY

We propose a compound method comprising three steps. We primarily utilize the item-to-item collaborative filtering algorithm, which matches each of the user's choices to the similar ones. It analyzes a class of item-based recommendation algorithms to present recommendations to users. Unlike the user-based collaborative filtering algorithm, the item-based approach looks into the set of items a target user has rated and computes how similar they are to a target item (i) , and then selects the (k) most similar items $(i_1, i_2, i_3..., i_k)$. It also simultaneously computes their corresponding similarities $(s_{i1}, s_{i2}, s_{i3}..., s_{ik})$. Once the most similar items are found, the recommendation is then computed by computing the weighted average of the target user's ratings on these similar items.

In the first step, our proposed method elaborates on social network distances and friendship information to help recommender systems customize the target user set. Friends are seen as more qualified to make good and useful recommendations compared to recommender systems primarily because they are assumed to know more about the recommendee. Memory based approaches make rankings recommendations for people depended on their previous rankings. Typically, the prediction is computed from the past usage of other users. To find people who are similar to a user, the system divides the users' friends (target set) into subsets, and treats the task as a social clustering problem. Clustering is then done on the social measures in the second step. Three of the measures used are common mutual friends, social graph distance, and relationship closeness. Recommendations are not made in rational isolation, which means that they are not evaluated merely by their information value; rather, they are delivered within an informal community of professional networks and social context.

In the second step, the clustering algorithm works on a socially worker set to build a job table and similar job pairs people who tend to do similar work. K-means is a rather simple, but well-known, algorithm for grouping objects and clustering. In this algorithm, all objects are represented as a set of numerical social network measures. The algorithm then randomly chooses k points in vector space to serve as the initial centers of the clusters. Subsequently, each object is assigned to the center to which it is the closest. Distance measure is chosen by the algorithms and determined by the social features. At the end of the target user selection process, an item table (matrix) and pairs are ready for processing from the recommendation algorithm.

In the third step, highly recommended items are defined by computing a similarity metric, such as cosine distance, on the vectors in the matrix table. In this way, item-to-item recom-

mendation can compute an ordered prediction for users. A crucial point for the item based collaborative-filtering approach is computation of the relationship between two elements i and j followed by selection of the most similar items $S_{i,j}$. It is possible to compute the similarity between two items in a number of ways; however, the most common method is to use the cosine measure, in which each vector corresponds to a company rather than a worker, and the vector's M dimensions correspond to persons who have worked at that company. Given a similar-company table, the algorithm finds similar companies for which the target people have worked, aggregates those companies, and then recommends the most popular or correlated companies. This computation is very fast as it depends only on the number of items corresponding to where the user worked.

In this method, the recommendation algorithm does not estimate the employee's preferences regarding jobs, like video rankings, yet tries to advise companies for which they may work. In this scenario, all of recommendations declared to the worker is probability-ordinated, this is favourable to evaluate proposed method over a range of recommendation stack lengths, instead of using limited certain length. In this stack, recommendations have normalized probability values. Thus, we can compute curves that compare recall to precision, or false positive rate to true positive rate. The curves of the previous version are identified commonly as the curves of precision-recall, while those of the next version are identified as ROC (Receiver Operating Characteristic curves). Information related to present and past employment data allows us to draw ROC curves and calculate RMSE values after recommendation.

In evaluating the recommendation system on test datasets, we must answer these questions; 'How to measure performance and which threshold to compare with'. We introduce three steps of our recommendation method, and mention where past study has concentrated. We have used the literature and developed our approach to design evaluation methodology and assemble the metrics used in performance calculation. Our approach is the comparison of parallel (ROC)curves. We understand that trial recommender suggest remarkably well; in some cases it surpasses more experienced algorithms. For threshold values of performance we advance the use of trial recommender; algorithm that are pointless to realize yet gain performance that is well above random too. The aim of our experiments is to find the most explanatory definition of performance for the developed algorithm and our proposed method.

As pointed out above, our three step methodology and its' algorithms have been proposed to make recommendations.Because of the validation of our recommendation and comparison of our methodology, the focus will be on a specific algorithm called restricted Boltzmann machines (RBMs). The importance of testing this particular model is actually two-induced. First, it was one of the best single model that has been used during the *Netflix* challenge. Every leading team included several variations of this model in their final blending. Second, its applications are not limited to recommender systems. They have been used for various other tasks, such as digit recognition, document retrieval.

The performance norm is also one of our research areas. Here, the best social network measure should maximize a predictive performance criterion as well as a computational perfor-

mance criterion. That is, we seek the best recommenders that are good at predicting items and efficiently compute over massive user datasets. There is no single method that works best on all given problems; therefore, we have to test the various similarity algorithms in order to compare their performances and to find the characteristics of data that determine the recommender on the dataset.

1.3 RELATED WORK

Many Collaborative Filtering methods [5],[4] base their recommendations on community selections (e.g., user ratings and purchase records), ignoring user and item property (e.g., demographics and product descriptions). On the other hand, solid content-based Filtering or information Filtering methods [51],[52] typically match query words or other user data with item property information, ignoring data from other users. Some hybrid algorithms combine both techniques [63],[5],[54]. Though 'content' most of the time refers to descriptive words associated with an item, we use the term more generally to refer to any form of item property information including, for example, the list of actresses in a film.

Likewise, early recommender algorithms were merely collaborative filters that computed pairwise similarities among users and recommended items according to a similarity-weighted average [3], [4]. Breese et al. [5] refer to this class of algorithms as memory-based algorithms. After the progressive work in the *Grouplens* project in 1994 [1], Collaborative Filtering (CF) soon became one of the most popular algorithms utilized in recommender systems. In this thesis, we use approaches such as item-based CF and *Amazon*'s recommender [6] for comparison purposes.

Several recommender systems that use collaborative filtering as a basis can be found in both academia and industry. In addition to *Amazon*'s recommender, several collaborative filtering methods such as *Netflix Prize* have been proposed [8], [9]. *Google News* is known to be using collaborative filtering to recommend news articles to its users [10]. Further, although no details have been published officially, it is known that several Web 2.0 companies such as *Del.icio.us* [11], *Yelp* [12], and *Last.fm* [13] also use collaborative filtering based systems, at least as a part of their services. In academia, *Tapestry* [14] was one of the first systems developed that used collaborative filtering. *Tapestry* was not an automated system, it expected each user to identify like-minded users manually [15]. [16] gives a good overview of some well-known collaborative filtering recommendation systems in use in the e-commerce field.

One of the factors that has contributed the most to the recent availability of large streams of data is the explosion of social networks. Recommender systems have also jumped onto this new source of data [64]. The practical model that proved quality in the Netflix Prize was the Restricted Boltzmann Machine(RBM). RBM's can be understood as the fourth generation of Artificial Neural Networks. Restricted Boltzmann Machines(RBM) can be stacked to form Deep Belief Nets(DBN). For the Netflix Prize, Salakhutditnov et al [47]. For instance, social network data can be an efficient way to deal with user or item for the learning part of RBM.

Social information can, i.e., be used to select the most informative and relevant items or users to learning part of dataset. RBM algorithm can use this dataset to reveal the hidden and visible units in few steps. In any case, it seems clear that at the very least social trust can be used as a way to generate reason and computability that have a positive impact on the success of RBMs' energy function [65].

Moreover, the use of Restricted Boltzmann Machines(RBMs) for data processing has been popular in recent times, possibly due to the recent advances in efficient inference and learning. Social-based recommendations can also be combined with the more traditional content-based approaches and artificial neural network implementation [66]. As a matter of fact, social network information can be efficiently included in a pure recommender setting by, for instance, including it in the matrix factorization objective function and trigger energy function [65].

In another research, Salakhutditnov et al show how a class of two-layer undirected graphical models, called Restricted Boltzmann Machines(RBM's), can be used to model tabular and socially related data, such as user's ratings of movies. they present efficient learning and inference procedures for this class of models and demonstrate that RBM's can be successfully applied to the Netflix data set, containing connections between users and over 100 million user/movie ratings. Salakhutditnov demonstrates that deep generative models(an improved model of RBM) that contain many layers of latent variables and millions of parameters can be learned efficiently, and that the learned high-level feature representations can be successfully applied in a wide spectrum of application domains, including information retrieval, and regression and classification of social networks. Alternatively, he shows similar methods can be used for nonlinear dimensionality reduction [67].

Analysis and fusion of social measurements is important to understand what shapes the public's opinion and the sustainability of the global development. After all, modelling data collected from social responses is challenging as the data is typically complex and heterogeneous. The responses are a mixture of data types including binary, categorical, multi-categorical, continuous, ordinal, count and rank data. The challenge is therefore to effectively handle mixed data in the a unified fusion framework in order to perform inference and analysis. The research introduces eRBM (Embedded Restricted Boltzmann Machine) - a probabilistic latent variable model that can represent mixed data using a layer of hidden variables transparent across different types of data [68].

1.4 CONTRIBUTION AND ORGANIZATION

Our proposed collaborative filtering algorithm, which uses social network measures to improve and scale recommendations, makes three primary research contributions:

- Item prediction is typically calculated using information about past usage by other users. Like the most popular applications, our dataset contains information associated

with more than 100,000 users for prediction. Social network measures facilitate the creation of target user sets and subsets defined via these features. Firstly, the measures and calculation methods used are explained.

- Each of these social network distances assigns a numeric score to a user and k-Means cluster scores. After user clustering, the hybrid system prepares item (job) vectors for similarity calculations. Thus, secondly, the clustering method and preparation of collaborative filter inputs are detailed.

- The collaborative filtering algorithm generates items to recommend by computing distance metrics on job vectors. In this way, item-to-item recommendation can calculate ordered prediction for users. We also search for success among four different distance algorithms at this point: Cosine Distance, Euclidean Distance, Manhattan Distance and Minkowski Distance. Therefore, thirdly, we discuss item-to-item recommendation and distance (similarity) measures.

We discover three formulations of a pre-computed model of collaborative filtering to scale and develop the success of item based recommendation. Evaluation of formulations is an experimental testing of the quality of some various item based collaborative method and various social network measures.Our item based approach and various subtasks of it is introduced and detailed. Additionally, this thesis presents the results of various experiments, our methodology, detailed dataset, evaluation approach, discussion and analysis of evaluation and the results. Last comments are presented at the final chapter.

The remainder of this thesis is organized as follows: Chapter 2 provides background information pertaining to recommender systems and discusses various approaches used to implement them. In particular, we give a formal definition of the recommendation problem, and discuss existing approaches while paying special attention to collaborative filtering. Finally, we end the chapter with a brief introduction to social graphs.

In Chapter 3, we give details of the first phase of our research. We present the aim of our research and the methodology followed and, after presenting the details of the algorithms used, we discuss the empirical results obtained by us.

Chapter 4 covers the second phase of our research. We begin by discussing another solution to the problem and another methodology. Consequently, we compare various approaches and present the results of experiments conducted.

Finally, in Chapter 5, we give concluding remarks and outline future developments pertaining to the work presented.

CHAPTER 2

BACKGROUND

2.1 RECOMMENDATION SYSTEMS

2.1.1 OVERVIEW

Personal experience is the most beneficial issue when we have to make choices. However, most of the time, we need to observe the experiences of people that we trust due to lack of personal experience about alternatives [17]. Recommendation systems simply try to model this meaningful action. They can be viewed as specialized information filtering applications that aim to provide a much smaller and manageable subset from a large collection of items in which users may possibly be interested.

Formally, recommendation systems try to maximize a utility function L that measures how related a given item is to a given user. Let U be the set of all users, and Y be the set of all items. For a given user $u \in A$ and a given item $y \in Y$, L can be defined as

$$L : UxY \rightarrow S \tag{2.1}$$

where S is the subset of Y that contains relevant items for u . In most cases, it is desired that S be a list that is ordered by relevance to users' preferences. The recommendation problem then becomes one of looking for valid subsets of Y for each user that maximizes that user's utility for each element in the set:

$$\forall u \in u, \forall y^{'} \in Y^{'}{}_{u}, y^{'} = \underset{y \in Y}{\operatorname{argmax}} L(u, y) \tag{2.2}$$

Recommendation systems make use of user inputs in order to define the utility of an item to a user, and have become extremely common. When viewing a product on *Amazon.com*, the store will recommend additional items based on a matrix of what other shoppers bought along with the currently selected item [6]. In the table, the similarities are depend on the mutual interests of the group of customers. For instance, in this representation, interaction

between purchased products by customers are similarity factor.(i.e., products A - B are related since wide distribution of the costumers who bought product A also bought B). In addition to similarity factor, the table contains values that display rates of similarity between exclusive items. The algorithm produces personalized recommendations in this way : it fetches from the table set of similar products complementing to the products seen to be of concern of the customer. It properly merges and sorts the similar products in one set using the similarity values. and filter to produce a sorted set of recommendations. In addition, algorithm is revealed that to produce recommendation by using different approaches for using previous or presently the contents of customer's e-commerce choices.

In the literature, recommendation systems are classified into three categories [15]: content-based, collaborative filtering, and hybrid methods.

- Content-based recommendation systems try to find similar items to those in which a given user is known to be interested. Item descriptions, user comments, and other contents that describe items are used during similarity calculations.
- Collaborative filtering recommendation systems try to find users with similar tastes.
- Hybrid methods recommendation systems simply combine content-based methods with collaborative filtering methods.

Content-based recommendation systems and hybrid methods recommendation systems are beyond the scope of our work. Therefore, we present introductory information only about systems that use collaborative filtering [18] [5].

2.2 COLLABORATIVE FILTERING

Collaborative filtering(CF) aims at learning predictive models of user preferences, interests, or behavior from community data; that is, from a database of available user preferences. Thus, collaborative filtering can be viewed as a recommendation technology that aims to learn a user's tastes based on a community's previous actions. Of course community covers the user, thus collaborative filtering makes use of a user's previous actions as well. The main assumption of collaborative filtering is that those who agreed in the past tend to agree again in the future.

M. Balabanovic and Shoham [18] defined two main approaches to recommender systems: collaborative filtering and content-based filtering. In collaborative filtering, the recommendation is based on analysis of similar users to indicate items of preference. In content-based filtering, the recommendation is made through analysis of similar items for user has already realized and evaluated.

In this type of filtering, recommendations are made based on predictions of user preferences, resulting in interactions between other users. This type of filtering usually offers a higher

degree of surprise to the user with good recommendations and, in some cases, may offer totally irrelevant contents. Collaborative filtering systems, on the other hand, try to include people in the filtering process, since they can better assess documents than any computer task [17].

A first approach to this type of filtering [17] establishes recommendations based on items consumed by users with the same consumption pattern as the current user (see Figure 2.1).

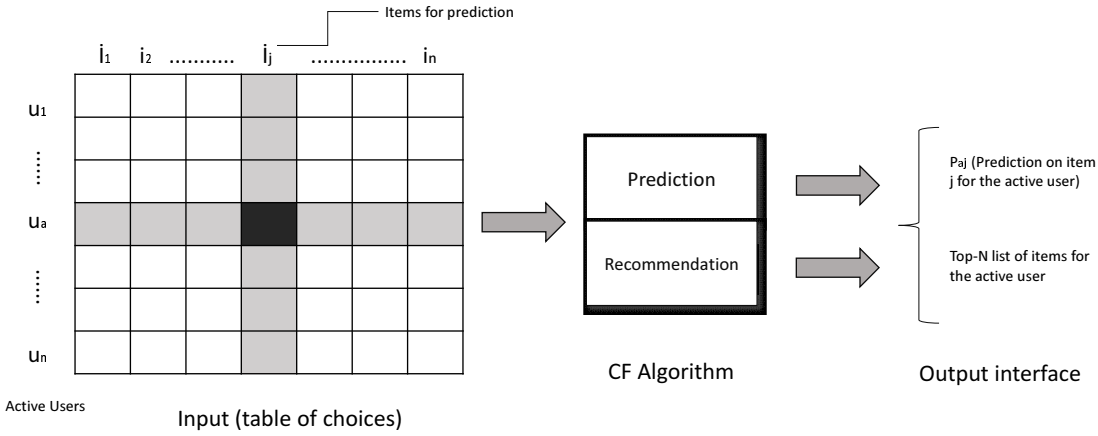


Figure 2.1: Collaborative Filtering demonstration

It is primarily used in e-commerce systems, such as those of *Amazon* and *Submarino*. To make this type of approach easier to understand, let us consider a user and a set comprising those users whose buying patterns are most similar to s (since they bought some of the same products x has bought). Now consider vectors (p, n), where p is a product and n is the number of times this product was purchased by a . If the vector set (p, n) is sorted in s descending order by n, the result will be an order for product recommendation for x. A variation may apply different weights to users, based on their relation to x users.

In the approach related to collaborative filtering, it is possible to solve the problem found in the recommendation approach using contents, where the user only receives items with similar contents. However, this approach does not solve other problems, such as the insertion of new items that will only be recommended after a certain number of users have read and assessed them into the base. Another issue is the handling of users who do not have interests that are similar to those of other members of the population. Thus, such unique users will not have peers on which the collaborative recommender system can base itself.

We can define collaborative filtering using the same formal notation that we used to define the recommendation problem. Collaborative filtering uses other users' utility functions $L(u', y)$ to predict the value of the utility of a user $L(u, y)$:

$$L(u_1, y), L(u_2, y).....L(u_m, y) \rightarrow L(u, y) where u_i \in M(u) \tag{2.3}$$

Consequently, $L(u, y)$ is a combination of several other users' utility functions. This is the collaboration part of collaborative filtering. Note that a new function M is introduced in Equation (2.3). M is the filtering part of collaborative filtering. It is responsible for selecting a high quality subset of all users. By the term high quality we refer to u relying on the tastes of users selected by M [19].

2.3 ITEM-TO-ITEM COLLABORATIVE FILTERING

The system utilizes an item-to-item collaborative filtering approach in making its job recommendations. This essentially means that for each job v_i, we build a neighbourhood of friend subset S_j at each step. Related items are taken from work history, when and where you worked at a job, the system that recommends you a job tends to work from that list. To determine the most similar match for a given item, the algorithm builds a similar-items table from the previous job pairs and vectors. We can build a job-to-job matrix by iterating through only those previous job pairs and computing a similarity metric for each pair.

This approach examines the set of items a target user has rated, computes how similar they are to the target item (i), and then selects the (k) most similar items ($i_1, i_2, i_3..., i_k$) [1]. In addition, their corresponding similarities (s_{i1}, s_{i2}, s_{i3}..., s_{ik}) are also simultaneously computed. Once the most similar items are found, the prediction is then computed by taking the weighted average of the target user's ratings on these similar items [1].

It is possible to compute the similarity between two items in different ways, but a common method is to use the cosine distance we mentioned earlier, in which each vector corresponds to a company rather than a worker, and the vector's M dimensions correspond to persons who have worked that job.

There are two main categories of objects which we specify as item and user, in the recommendation systems. User has choices for fixed item and we must extract those choices to the data-structure cautiously. A utility matrix is that data-structure and the data contains those item-user couples, a number that describes details of level of choice of item-user affinity.

The first question we must deal with is how to measure the similarity of users or jobs from their rows or columns in the utility matrix. There are a number of different ways to compute the similarity between jobs. Here, we present four such methods: Cosine Distance-based [20] similarity, Euclidean Distance-based [21] similarity, Manhattan Distance based [22] similarity, and Minkowski Distance-based [23] similarity.

2.3.1 COSINE DISTANCE

The cosine distance is a measure of distance between two vectors of an inner product area that measures the cosine of the angle between them. The cosine of 0° is 1, and it is less than 1 for any other angle [55]. It is thus a intuition of direction and not magnitude: two vectors with the same direction have a Cosine distance of 1, two vectors at 90° have a distance of 0, and two vectors diametrically opposed have a distance of -1, separate of their magnitude [55]. Cosine distance is particularly used in positive area, where the outcome is neatly bounded in [0,1].

In addition, note that these bounds implement for any number of dimensions, and Cosine distance is most commonly used in high-dimensional positive areas [55]. For example, in Information Retrieval, each term is notionally assigned a different dimension and a document is characterised by a vector where the value of each dimension corresponds to the number of times that term appears in the document. Cosine distance then gives a useful measure of how similar two documents are likely to be in terms of their subject matter [55]. In addition, it is used to measure cohesion within clusters in the field of data mining [28].

Also known as vector-based similarity, this formulation views two items and their ratings as vectors, and defines the similarity between them as the angle between these vectors. Two items are thought of as two vectors in the M dimensional user-space. Formally, in the ratings matrix, the similarity between items i and j, denoted by d, is given by

$$d = \cos(\vec{i}, \vec{j}) = \frac{\vec{i} \cdot \vec{j}}{||\vec{i}||_2 * ||\vec{j}||_2} \tag{2.4}$$

2.3.2 EUCLIDEAN DISTANCE

Very often, especially when measuring the distance in the plane, we use the formula for the Euclidean distance. The source of this formula is the Pythagorean theorem. In general, the distance between points x and y in a Euclidean space $\mathfrak{R}^2$ is given by

$$d = ||x, y|| = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2} \tag{2.5}$$

The euclidean distance is most often used to measure profiles of respondents across variables. For example, suppose our data consist of professional information on a sample of individuals, arranged as a respondent-by-variable matrix. Each row of the matrix is a vector of m units, where m is the number of variables. We can evaluate the similarity (or, in this case, the distance) between any pair of rows. Notice that for this kind of data, the variables are the columns. A variable records the results of a measurement. For our purposes, in fact, it is useful to think of the variable as the measuring device itself. This means that it has its own

proportion, which determines the size and type of units it can have. For instance, the income measurer might yield units between 0 and 2 million, while another variable, the education measurer, might yield units from 0 to 30. The fact that the income units are larger in general than the education units is not meaningful because the variables are measured on different proportions. In order to measure columns we must adjust for or take account of differences in proportion. But the row vectors are different. If one case has larger units in general then another case, this is because that case has more income, more education, etc., than the other case; it is not an artefact of differences in proportion, because rows do not have proportions: they are not even variables. In order to compute similarities or dissimilarities among rows, we do not need to (in fact, must not) try to adjust for differences in proportion [61]. Hence, Euclidean distance is usually the right measure for comparing cases.

2.3.3 MANHATTAN DISTANCE

Named taxicab geometry, contemplated by H. Minkowski , is a mode of maths in which the usual similarity function or metric of Euclidean distance is replaced by a new metric in which the distance between two points is the sum of the absolute differences of their Cartesian coordinates. The taxicab metric is also known as rectilinear distance, L_1 distance norm (e.g. L^p space), city block distance, Manhattan distance, or Manhattan length, with corresponding variations in the name of the mathematics [56]. The latter names allude to the grid layout of most streets on the isle of Manhattan, which causes the shortest path a car could take between two crossing in the borough to have length equal to the crossings' distance in taxicab geometry [56].

The Manhattan distance, d, between two vectors x, y in an n-dimensional real vector space with a fixed Cartesian coordinate system, is the sum of the lengths of the projections of the line segment between the points onto the coordinate axes [56]. This number is equal to the length of all paths connecting $Point(x_1, y_1)$ and $Point(x_2, y_2)$ along the horizontal and vertical segments, without ever going back.

$$d = ||x, y|| = \sum_{i=1}^{n} |x_i - y_i| \tag{2.6}$$

2.3.4 MINKOWSKI DISTANCE

The Minkowski distance is a generalized metric that includes others as special cases of the generalized form. Although theoretically infinitely many measures can exist by varying the order of the equation, only three have gained importance.

$$d^{MKD}(i,j) = \lambda \sqrt{\sum_{k=0}^{n-1} |y_{i,k} - y_{j,k}|^{\lambda}} \tag{2.7}$$

In the equation d^{MKD} is the Minkowski distance between the data record i and j, k the index of a variable, n the total number of variables y, and λ the order of the Minkowski metric. Although it is defined for any $\lambda > 0$, it is rarely used for values other than 1, 2, and ∞. As infinity cannot be displayed in a computer's arithmetic, the Minkowski metric is transformed for $\lambda = \infty$ and becomes :

$$d^{MKD}(i,j) = \lim_{\lambda \to \infty} (\lambda \sqrt{\sum_{k=0}^{n-1} |y_{i,k} - y_{j,k}|^{\lambda}}) = \max_{k} |y_{i,k} - y_{j,k}| \tag{2.8}$$

The Minkowski distance is often used when variables are measured on ratio scales with an absolute zero value. Variables with a wider range can overpower the result. Even a few outliers with high values bias the result and disregard the alikeness given by a couple of variables with a lower upper bound.

This is the generalized metric distance. When $\lambda = 1$ it becomes city block distance and when $\lambda = 0$, it becomes Euclidean distance. Chebyshev distance is a special case of Minkowski distance with $\lambda = \infty$ (taking a limit). This distance can be used for both ordinal and quantitative variables.

2.4 CHALLENGES OF CONTENT FILTERING IN RECOMMENDATION SYSTEMS

Other than some research applications, it is hard to find multi-purpose recommendation systems, especially as an on-line product. Prediction methods and learning algorithms used in recommendation systems are highly customized to the service that is using the recommendation system. This is due to the nature of the datasets available. In fact, the performance of the algorithms presented in this research area is highly dependent on the dataset it is dealing with. Although there is no generic system, all the systems developed face a set of common problems: Data sparsity, new items added to the system, new users added to the system, and computational resource requirements.

Moreover, data sparsity is arguably the most problematic issue for all recommender systems. Usually, the number of ratings available to the system is very small compared to the number of ratings that the system is expected to predict [15]. In fact, data sparsity is one key issue that results in the recommendation problem itself. If we knew all the ratings then there would be no need to compute missing ratings. Because of data sparsity, it is very hard for a recommendation system to mimic users' tastes [15]. Some popular datasets that have been used by researchers are listed in Table 3.1. Unfortunately, real world datasets are frequently sparser.

The weakness of the nearest neighbor algorithm for large, sparse databases led us to explore alternative recommender system algorithms. We first attempted to bridge the sparsity in the approach by incorporating semi-intelligent filtering agents into the system [24][25]. These agents evaluated and rated each item using syntactic features. By providing a dense ratings set, they helped alleviate coverage and improved quality. The filtering agent solution, however, did not address the fundamental problem of poor relationships among like-minded but sparse-rating users. To explore that we took an algorithmic approach and used Latent Semantic Indexing (LSI) to capture the similarity between users and items in a reduced dimensional space [16][26]. In this thesis, we examine another technique, the item-to-item approach, in addressing these challenges, especially the scalability challenge. The main idea here is to analyze the item-item representation matrix to identify relations between different items and then to use these relations to compute the prediction score for a given user-item pair. The intuition underlying this approach is that a user is interested in purchasing items that are similar to the items the user liked earlier and will tend to avoid items that are similar to the items the user did not like earlier. These techniques do not require identification of the neighborhood of similar users when a recommendation is requested; as a result, they tend to produce much faster recommendations. A number of different schemes have been proposed to compute the association between items ranging from the probabilistic approach [5] to more traditional item-item correlations [27].

User to user recommender advantages: Web stores like *Amazon* have a large customer database more than their products. Hence, computation of finding similar products is easier than finding similar customers. Computation of finding similar customers is difficult task because specific customers generally have lots of different tastes, but specific products are member of less genres.

Item to item recommender advantages: This method is easier to realize and updates recommendations more quickly. When a new product is purchased, system might advise a new recommendation in the item to item recommender. though a Item to item recommender customer might have to wait until the recommendations have been re-computed. The item to item recommender may be more simpler influenced in different domains also, not only in the suggestions , but also in the 'similar items, another user purchased' section while viewing specific product.

Item to item recommender disadvantages: Recommender does not suggest much different kind of opportunities or diversity in item to item approach, so you might get boring recommendations

2.5 WORKING WITH SOCIAL NETWORK MEASURES

Social networking sites such as *MySpace* and *Facebook*, as well as the future internet enable people and companies to contact each other with human-friendly names. Today lots of In-

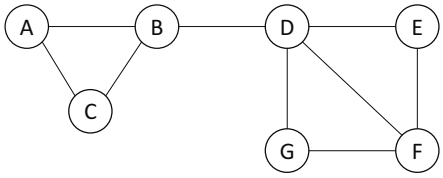


Figure 2.2: Example of a small social network

ternet users are using lots of social websites to stay connected with their friends, discover new 'friends', and to share user-created content, such as photos, movies, web bookmarks, and blogs, even through mobile platform help for cell phones [57]. Social network data are defined by actors and by relations (or 'nodes' and 'edges'). The nodes or actors part of network data would seem to be pretty straight-forward. Other empirical approaches in the social sciences also think in terms of cases or subjects or basic elements and the like. There is one difference with most network data, however, that makes a big difference in how such data are usually collected and the kinds of samples and connections that are studied.

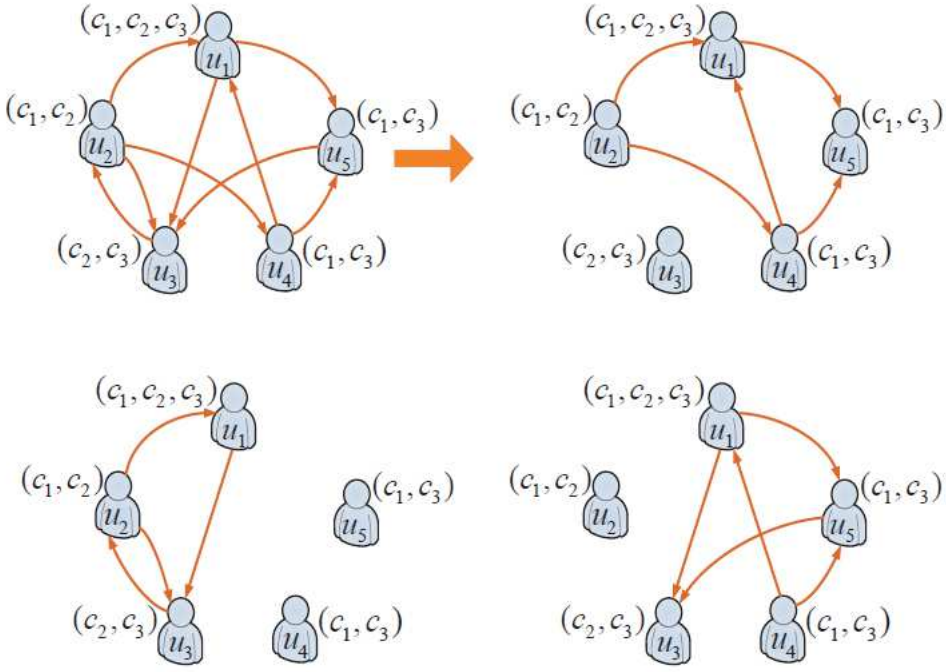


Figure 2.3: Illustration of inferred subgraphs, each user is separated with the sample distance measures respectively.

A social network is actually a kind of network that is representative of a broader class of networks called 'social'. One of the essential characteristics of a social network is an assumption

of non-randomness or locality [28]. This condition is the hardest to formalize, but the intuition is that relationships tend to cluster. That is, if entity A is related to both B and C, then there is a higher probability than average that B and C are related. The main idea behind social network measures tends to be this higher probability. Social networks are naturally modeled as undirected graphs. The entities are the nodes, and an edge connects two nodes if the nodes are related by the relationship that characterizes the network. If there is a degree associated with the relationship, this degree is signified by labeling the edges. Measures between A, B, and C can be calculated by counting the edges among them [28].

If we were to apply standard clustering techniques to the graph of a social network, our first step would be to define a distance measure. When the edges of the graph have labels, these labels might be usable as a distance measure, depending on what they represent.

2.5.1 MUTUAL FRIENDS

However, when the edges are unlabeled, as in a 'friends' graph, there is not much we can do to define a suitable distance. Our first instinct is to assume that nodes are close if they have an edge between them, and distant if not. Thus, we could say that the distance $d(v_1, v_2)$ is zero if there is an $edge(v_1, v_2)$ and one if there is no such edge. We could use any other two values, even one and ∞, as long as the distance is closer when there is an edge.

Links from user v_1 to v_2 are listed in the friends list of v_1. This list is intersected into a reciprocal subset for a user v_2, MF is the measure between v_1 and v_2. This value is grouped for clustering with other friends v_3...:

$$SetofT : T_{v_1} = \{edge(v_1, v_2).... \in T\} \tag{2.9}$$

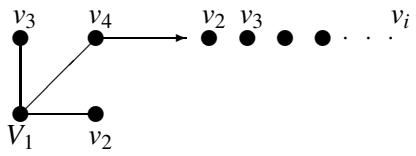
$$MF_{v_1,v_2} : |T_{v_1} \cap T_{v_2}| = \begin{Bmatrix} v_1, v_2 : v_1, v_2 \in T_{v_1} \\ \wedge \\ v_1, v_2 : v_1, v_2 \in T_{v_2} \end{Bmatrix} \tag{2.10}$$

$$ListMF : (MF_{v_1,v_2}, MF_{v_1,v_3}, MF_{v_1,v_4}..., MF_{v_1,v_i}) \tag{2.11}$$

2.5.2 FRIENDS' DISTANCE

Because there are problems with standard clustering methods, several specialized clustering techniques have been developed to find communities in social networks. In this step, we model the network as a graph. It can be thought of as a finite set of vertices (friends) $\{v_1, v_2, ..., v_n\}$ and a set of relations between each pair of vertices. Each of the pairs $\binom{n}{2}$ of vertices is either adjacent or nonadjacent. If a pair of vertices is adjacent, we say there is an edge connecting the two vertices, and they are called neighbors. In graphs, a walk (distance) of length k is a sequence of vertices $\{v_1, v_2, ..., v_k\}$ such that v_1 is adjacent to v_{i+1} for each i:

$$walk : v_i \sim v_{i+1} \forall i \tag{2.12}$$



The depiction above illustrates this concept, in which we are looking at vertex (friend) V_1; actually each vertex, 1 through k, is part of its own sub-graph. Friend t can be adjacent to i of the remaining $n-k$ friend, and i can range from zero to $n-k$. If we now sum the number of trees we get from adding each vertex 1 across all possible values of i, we can obtain a value for $FD_{n,k}$. Formally we have

$$FD_{n,k} = \sum_{i=0}^{n-k} \binom{n-k}{i} FD_{n-1,(k-1)+1} \tag{2.13}$$

$$ListFD : (FD_{v_1,v_2}, FD_{v_1,v_3}, FD_{v_1,v_4}..., FD_{v_1,v_i}) \tag{2.14}$$

2.5.3 FRIENDS' MUTUAL FRIENDS

Another method of showing that a set has a certain number of common friends is to find an intersection between that friend's set and the set of other friends with a known shared set of elements. In this case, we find all the elements in the intersection between the set of sequences and the set of friends:

$$SetofT : T_{v_1} = \{edge(v_1, v_2).... \in T\} \tag{2.15}$$

$$FMF_{n,k} = \sum_{i=0}^{n-k} \binom{|T_{v_n} \cap T_{v_k}|}{i} T_{n-1,(k-1)+1} \tag{2.16}$$

$$ListFMF : (FMF_{v_1,v_2}, FMF_{v_1,v_3}, ..., FMF_{v_1,v_i}) \tag{2.17}$$

2.5.4 WEIGHTED FRIENDS

In contrast to treating direct friends mutually, as in MF, WF indicates that each connection between friends has particular impact(or weight) relatively. Hence we can calculate the weight of connection(edge) between target user and friend by measuring the connection closeness. If a friend has higher impact from a ordinary friend, item of target user is closer choice for that user. So, the probability of the choice is proposed to the gained impact in each usage rate. We also call that approach related friends:

$$Setof T : T_{v_1} = \{edge(v_1, v_2).... \in T\} \tag{2.18}$$

$$WF_{v_1,v_2} : Pr(Rv_1 = k|\{Rv_2 = rv_2 : \forall W \in M(V) \cap V(I)\}) \tag{2.19}$$

$$WF_{v_1,v_2} = \frac{1}{Z}\Sigma_v w(V, I)\delta(k, rv_2) \tag{2.20}$$

$$ListWF_{:}(WF_{v_1,v_2}, WF_{v_1,v_3}, WF_{v_1,v_4}..., WF_{v_1,v_i}) \tag{2.21}$$

z is the constant of normalize and connection weight between user and friend is $w(V, I)$. In equation 2.20, cosine distance is used to measure base weight between v_2, v_1. $\delta(k, rv_2)$ is a delta function that returns one only when $\delta rv_2 = k$, and zero otherwise. WF is nearly the same as a relational-neighbor classifier [29], which works very effective in the clustering of relational datasets such as citations and items.

CHAPTER 3

PROPOSED RECOMMENDATION SYSTEM

In this chapter, we discuss the first part of our research, in which we improve the popular item-based collaborative filtering algorithms, and propose to solve the collaborative filtering performance problem in by taking the advantage of social graph data. We also empirically compare the recommendation performances of new algorithms with those of their original forms.

In the first section, we discuss the aim of this research and the methodologies followed. In the second section, we discuss the datasets used in detail. In the third section, we give details of algorithmic modifications and inspect each algorithm used separately. Finally, we close this chapter with a review of the evaluation methods used and the results obtained.

3.1 AIM AND METHODOLOGY

For virtually all collaborative filtering methods, a very sparse rating matrix is provided on which basis accurate predictions are expected to be made. Some of these methods look for implicit features, while others look for collaboration patterns. However, all of them meet at the same point: if the training set is of a high enough quality, more accurate predictions can be provided. In pure collaborative filtering applications, the dataset is problematic because it only contains positive examples. In this part of our research, we introduce social graphs as a new data source for classical approaches and examine ways in which this data can be utilized. Using social graph data in accordance with features of datasets, we believe that the performance of item-to-item recommendation methods can be improved.

After the presentation of the abstract bases of both recommendation systems and social networks, the proposition of recommendation system for social network will be described. First, the overall problem definition of recommendations for social network is provided. After that, the structure of system, as well as the concept of recommendation for social networks are described. Furthermore, user profile that contains all data needed for recommendation and the recommendation algorithm is presented. The proposed algorithm consists of the main elements, which are: clustering the target user network, K-Means clustering on social distances, collaborative filtering as recommender. Finally the definition of social recommendation pro-

cess that respects users' social connections and evolution results is provided.

Further, in this section, we introduce the dataset used in our study , and demonstrate many interesting features of this dataset. Our dataset is from a real on-line social network *Linked.com*. As one of the most popular Web 2.0 websites, *LinkedIn* connects people to their trusted contacts and helps them exchange knowledge, ideas, and opportunities with a broader network of professionals [30]. Users that come to this site can either look for information from *LinkedIn* or make their own voices heard by writing reviews for some local commercial entities with which they have experience. *Yelp* provides a user-page for each local member entity. At the top of each user's homepage is a profile of the user, which includes user attributes such as interests, specialities, groups, and education information. In addition, the homepage contains a list of reviews of users who have worked with the user in the past [29],[62].

3.2 PROBLEM DEFINITION

To make recommendation in a social network is different from general recommender methodologies, because the main object is human beings and social recommender advise to a realistic object rather than an inorganic object. Then , a worker starts a friendship with a co-worker, but he/she may reply either negatively or positively to the invitation afterwards. Items, contents or business impossibly can foster that cooperation. Furthermore, the friendship between two people is bidirectional, but relationship between worker and job is unidirectional. Hence, the goal of such recommendation system is to find for the current user reaction by rummaging out other people who would also be likely to act in a benign way. Items do not have any free will and cannot ignore to be selected. In contrary, one worker can refuse to keep up the relationship with another worker. Since, the recommendation in a social network require to comply with professions and preferences of workers. As as result, our recommender algorithm should advise to worker only fellows of social network, whom would be possibly the good co-workers of friends for this worker.

Nevertheless, the social network leaders can accomplish their own practice. So, they firstly aim to construct the closest social network within many strong relationships as much as possible. Another desirable intent could be to construct the social network where the connections mirror the strong relations between the network and users. In this way, the social network might contain lots of heterogeneous groups . Separate type of recommendations are able to aggravate the progress of the social network. The main goal is to reach the community where the relations between people are constant in any case. To achieve the chosen goal of the owners of specific social network the concepts we presented as social network distances, which were described in Chapter 2.6.

The common intent of the propositioned below framework is make possible the modification of recommendations to the society of the specific worker as well as to the common practice of the social network.

3.3 DATASET

Like other data mining fields, recommendation systems also have standard datasets that are used by researchers to compare the performance of their methods. Some of the standard datasets are listed above. Unfortunately, in our research we were unable to use any of these standard datasets because none of them contains information on the social relationships between its users. To test the effect of using social data we needed a dataset that contains ratings as well as user relations.

Table3.1: Datasets that have been used to test recommendation systems.

Dataset	User Count	Item Count	Rating Count
MovieLens	943	1,682	100,000
Netflix	480,189	17,770	100,480,507
EachMovie	72,916	1,628	2,811,983

Our dataset is a specific database comprising the social network information of business people in Turkey. It stores the professional profile of each user, lists their present and past employment, and links them to former and current colleagues and classmates. It also has detailed information about companies and their properties. Because there are no publicly available datasets for experimentation that include both friendships and professional history, we collected live data from one of the most prominent social network sites *LinkedIn*.

Our detailed database was begun collecting in fall 2010. It has information on over 100,000 workers, comprising their professions and careers, and over 70,000+ different companies. We randomly selected different worker subsets that include more than 1000 social workers each having more than 20 friends.

3.3.1 DATASET STRUCTURE AND CONSTRUCTION

The first step in our measurable evaluation is collecting data sets that we can use to evaluate the recommendation algorithms we will form to address this recommender task. Conductive to test our algorithms, we must need to use data sets that are practical representations of the daily operation of a professional social network.

As we know, [24] are the only ones to have clearly discussed the challenges of creating and using datasets specifically for testing recommender algorithms. They worry about that it is very critical that the tasks your algorithm is designed to help are similar to the tasks supported by the system from which the data was collected [24]. An example would be the *MovieLens2* film recommender system, which offers a solution for the 'Find Good Items' user task. This means that the recommender system tends to show those items it is most satisfied about,

subsequently resulting in more ratings for these acceptable and often more popular films. It is critical to use a selection that is typical of the application in terms of the number, size, and type of resources. This reflects the point made when describing the first issue in the beginning of this section: an acceptable data set is a practical representation of actual content declared in the professional social network. Besides, advices should be typical of the real-world application. In our recommendation system, advices correspond to the worker profiles declare on the professional social network one user profile is matched against all dataset just as single advices is matched against the job selection. We therefore recommend collecting a practical and sufficiently large set of employee profiles in order to create an acceptable recommendation data set.

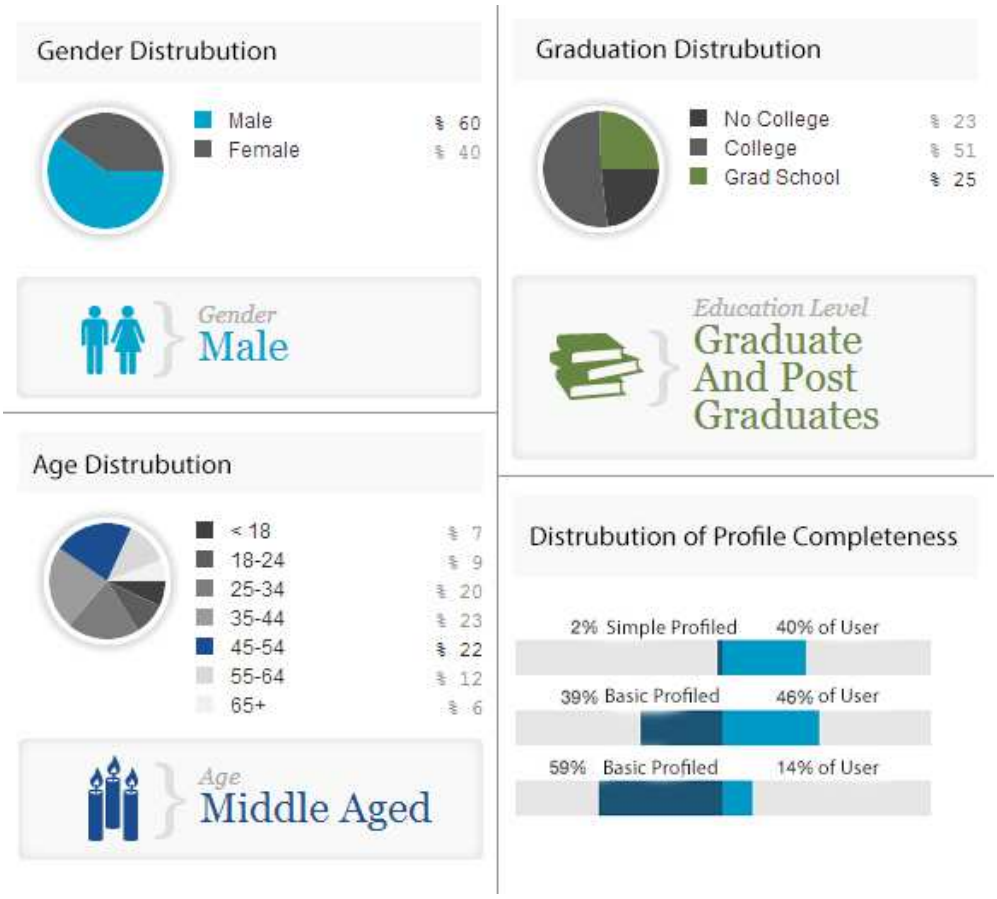


Figure 3.1: Demographics and Statistics of Dataset.

In addition to maintaining the general characteristics of the recommendation system, *LinkedIn* can draw attraction of more employees because of having professional characteristics of socialization. Particularly, *LinkedIn* lets employees to create their own network by requesting their co-workers to join or make new relationships from users presenting in social network. Friendships in the network is a mutual connection. It purposes that whether a user accepts another one as a co-worker, the first one is repentantly accept as a co-worker of the second one.

Profile page of worker includes the reviews added by the co-worker, simple profile info, and contacts to the co-workers that are mutually accepted by this worker. In today's professional world, people change jobs and locations constantly. Their contacts update all their profiles, keeping them current with their latest jobs, projects, and contact information. Consequently, they stay in close contact with great tools to communicate and collaborate.

Figure 3.1 lists some of the basic statistics of the raw, plain data sets we collected. For each of the three dataset selection, we list the distribution percentage in terms of age, education, gender, and profile completeness. Likewise, we show the average percentage of the selection of one subset type to another. For instance, the average percentage of profile completeness for dataset is equal to 39, while the maximum numbers of users that give all information about their profile in dataset is 346 in one of our raw sample data set. We do not list minimum counts for any of the calculated measures, because they are always equal to 0 or 1, depending on the measure.

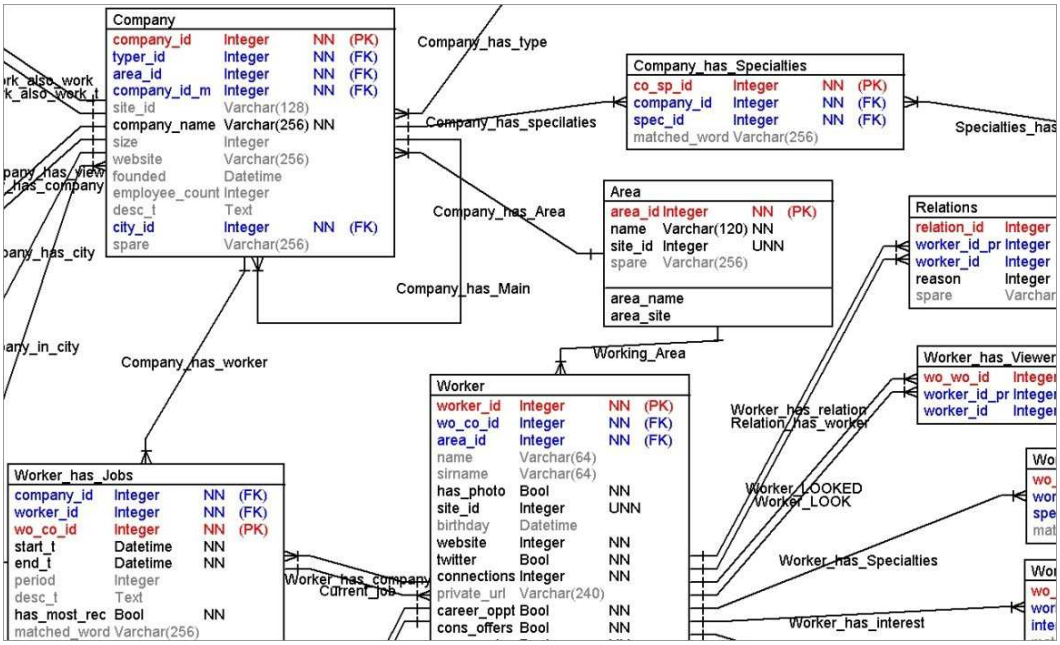


Figure 3.2: Corresponding graphical representation of SQL schema in which each table represents a Worker-Company relation in the dataset, tables are connected by foreign keys, and workers have friend relations in 'Relation'.

Because the profile page is the most important data source on *LinkedIn*, our study domain is profile pages of workers. At first, We crawled,fetched and parsed the profile pages of users in Turkey that could be searched via *Google* and *Yahoo* search engines, and accumulated close to 300,000 profiles. By tracing the company links in the profile pages of users we were also able to crawl the homepages of all related companies, which resulted in a total of 80,000 companies. On the basis of the friends' contact in each profile page of user, we could find

connections of the crawled profile pages of users. Hence we can rebuild the professional social network. Indicatively the relations we gathered for each worker might be a sub-set of the current relation that existed on her/his profile. That is the reason all workers in the dataset contain at least one peace of information. Conversely, the data and network that we fetched focused on the professional network.

To demonstrate workers' jobs and their professional relationships, an SQL schema is constructed and represents each worker in it. Because friends in our dataset are mutual, the structure of the social network is that of an undirected graph. In the schema, a user is referred to as a worker and their friends are stored in a 'relations' table.

3.3.2 PROFESSIONAL NETWORK ON DATASET

Business networking is a socio-economic activity in which groups of like-minded business people recognize, create, or act upon business opportunities. A business network is a type of social network whose primary reason for existing is business activity. There are several prominent business networking organizations that create models of networking activity that, when followed, allow business persons to build new business relationships and generate business opportunities at the same time. A professional network service is an information technology implementation in support of business networking. As an example, a business network may agree to meet weekly or monthly for the purpose of exchanging business leads and referrals with fellow members. To complement this activity, members often meet outside this circle, on their own time, and build their own one-to-one relationship with fellow members.

Table3.2: Dataset analysis used for our recommendation system(social connections)

Random Selection	User Count	Friendship Count	Average Number of Friends
Selection 1	1000	223.321	31
Selection 2	300	89.300	38
Selection 3	750	190.234	34
Total Distribution	121.131	4.527.263	32

Business networking can be conducted in a local business community, or on a larger scale via the Internet. Our dataset is an example of that kind of topology. We obtained the following statistics from the introductory analyses on the dataset. Each dataset worker's data is 39% complete on average and each company on average has 39 workers. The number of crawled users is 321,211 and the number of fetched users is 121.341 in the data-set. In terms of friends, the average number of directly connected friends of every user is 32. As shown in Table 3.2, the relationship between count of friends and count of users demonstrates that it nearly keeps a power law distribution. So, while a few users have 200-300 friends, most workers have 20-30 friends.As shown in Table 3.3 the same demonstration also covers the

relationship between workers and jobs. Because most users on *LinkedIn* generally work in IT companies, the sparsity of the dataset is expected to be acceptable. Substantially, the sparsity of this dataset(e.g., the percentage of worker/job pairs whose profession is ambiguous) is 69.86%.

Continuously on the dataset the following study is performed by addressing classification of employment. Our aim is to find a solution to these questions. Relevant workers(friendship) tend to choose same jobs more than irrelevant ones(non-friendship)? Relevant workers(friendship) tend to work in the same jobs more than irrelevant ones(non-friendship)? Definitely, investigating these problems are crucial in a recommendation system.

Table3.3: Dataset analysis used for our recommendation system(user-item pairs)

Random Selection	User Count	Job Count	Average Number of Job
Selection 1	1000	4334	4,3
Selection 2	300	1452	3,7
Selection 3	750	3578	3,4
Total Distribution	121.131	32.105	4,1

3.4 CLUSTERING THE TARGET USER NETWORK

One important aspect of social networks is that they contain specific subsets of entities that are connected by many edges. These correspond to groups of friends in school or groups of communities that are interested in the same expertise. In this section, we consider clustering of the graph as a way to identify communities using social network measures.

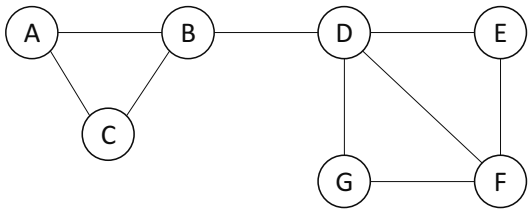


Figure 3.3: Repeat of Figure 2.2

If we were to apply standard clustering techniques to a social network graph, our first step would be to define a distance measure. When the edges of the graph have labels, these labels might be usable as a measure, depending on what they represent. However, when the edges are unlabeled, as in a friends graph, there is not much we can do to define a suitable distance.

Our first instinct is to assume that nodes are close if they have an edge between them, and distant if not. Thus, we define the weight $d(x, y)$ as a numeric value if there is an $edge(x, y)$ calculated by social measures and zero if there is no such edge. We could use any other two values, even one and ∞, as long as the distance is closer when there is an edge.

There are two general approaches to clustering: hierarchical (agglomerative) and weight-assignment. When considering how each of these would work on a social-network graph, first, consider the hierarchical methods; in particular, suppose that we use the minimum distance between nodes of the two clusters as the cluster subset distance. Hierarchical clustering of a social network graph starts by combining two nodes that are connected by an edge. Thinking successively, edges that are not between two nodes of the same cluster would be chosen randomly to combine the clusters to which their two nodes belong. The choices would be random because all distances represented by an edge are the same. However, we use weight-assignment because of its randomness and the fact that it does not represent all the edges at the same time [28].

Because there are problems with standard clustering methods, specialized clustering techniques have been developed to find specific subsets in social networks. We consider a simple one that is based on giving the edges betweenness scores that are least likely to be inside a subset. We define the *betweenness* of an $edge(a, b)$ to be the number of pairs of nodes v_1 and v_2 such that the $edge(a, b)$ lies on the social network measures between v_1 and v_2. More precisely, since we explained the difference in betweenness scores between v_1 and v_2, $edge(a, b)$ is credited with the fraction of those scores that include the $edge(a, b)$. It suggests that the $edge(a, b)$ runs between two different subsets; that is, a and b do not belong to the same community.

The aim of the recommender algorithm is to discover either person b ought to be recommended to user a. It is accomplished by using the formulation of similarity $r(a \Rightarrow b)$

For example, Mutual Friends is links from user v_1 to v_2 are listed in the friends list of v_1. This list is intersected into reciprocal subset for a user v_2, MF is the measure between v_1 and v_2. This value is grouped for clustering with other friends v_3... :

$$SetofT : T_{v_1} = \{edge(v_1, v_2).... \in T\} \tag{3.1}$$

$$c(a, b) = MF_{v_1,v_2} : |T_{v_1} \cap T_{v_2}| = \begin{Bmatrix} v_1, v_2 : v_1, v_2 \in T_{v_1} \\ \wedge \\ v_1, v_2 : v_1, v_2 \in T_{v_2} \end{Bmatrix} \tag{3.2}$$

$$\tag{3.3}$$

$$r(a \Rightarrow b) = \sum_{j=1}^{N} \beta c(a, b) \tag{3.4}$$

- $c(a, b)$ introduces the social behaviour of the worker as part of made connections and it is the correspondent of formulation of relation starting

- β is employed to make possible the computation of the social network and triggers the calculation. E.g.,then the aim of the social network is to construct the social network richness despite of the immediate similarity of workers respectively, if $c(a, b)$ is in upper level. The number of this fixed constant strongly rely upon the practice.

The betweenness scores for the edges of a graph are analogous to a distance measure on the nodes of the graph. It is not exactly a distance measure because it is not defined for pairs of nodes that are unconnected by an edge. However, we cluster by taking the edges in order of increasing betweenness and add them to the graph one at a time. At each step, the connected components of the graph are clustered using K-Means cluster. The higher the betweenness we allow, the more edges we get, and the more specific the clusters become.

The thing that should be asserted is that the tie between two persons is generally the combination of many different kinds of relationships, which can differ in distance. For Instance, we can assume the theoretical situation of people who are employed in a company. They are not only co-workers, but also other relationships exist between them. E.g., although person *D* and *G* work in the same company, they are not co-workers, but friends. The reality that two people are employed in the same company does not mean that they work together. Notably in big company, people can work in different departments and even do not know each other.

Furthermore, each of the relationship can differ in order. For example, person *a* can claim that is a really good friend of *b*, whereas *a* can admit that *b* is a friend but not so close as *a* thinks. The connections can also differ in their distance, e.g. the reality that *a* and *b* are neighbours can be less away than the reality that they are co-workers. It is possible that they work together every day, but they do not speak with each another out of company.

3.5 ALGORITHMS

Several algorithms have been proposed in academia to solve the collaborative filtering problem. We chose three popular algorithms to focus our research on the value of social graphs for our collaborative filtering problem rather than on algorithmic details. Choosing well-studied algorithms also makes us more confident about the robustness of the algorithms and the results they produce. In fact, several algorithms that have been proposed perform well on a specific dataset but are hardly tolerable to any dataset changes.

Although we chose popular algorithms, our results are not comparable to those of previous research because we were unable to use a standard dataset. However, the behaviors of the algorithms (in terms of convergence rates, running times, memory consumption, etc.) were as expected in our test runs.

This section covers the details of the standard algorithms used and the modifications we made to them. We start our discussion with a de facto clustering algorithm: K-Means. In the second part, we introduce detailed implementation of item-to-item collaborative filtering.Eventually in the third part, we discuss full composition, which is a method that we also try to explain in other parts of this thesis. Finally, we discuss how success of an algorithm can be evaluated.

3.5.1 K-MEANS

The K-Means algorithm is beyond doubt the most well-known algorithm in the machine learning field. This algorithm is simple to implement and produces acceptable results for different datasets. K-Means clustering generates a specific number of disjoint, flat (non-hierarchical) clusters. It is well suited to generating globular clusters. The K-Means method is a numerical, unsupervised, nondeterministic, and iterative procedure [2] and provides a simple and easy way to classify a given dataset by giving a certain number of clusters (assume k clusters) fixed priority. The algorithm partitions (or clusters) N data values into K disjoint subsets S_j containing N_j data values so as to minimize the sum-of-squares criterion:

$$J = \sum_{j=1}^{K} \sum_{n \in S_j} ||x_n - \mu_j||^2 \tag{3.5}$$

where x_n is a vector representing the n^th data value and μ_j is the geometric centroid of the data values in S_j. In general, the algorithm does not achieve a global minimum of J over the assignments. The algorithm consists of a simple re-estimation procedure. The data values (centroids) are assigned at random to the K sets. In step 1, the centroid is computed for each set. In step 2, every value is assigned to the cluster whose centroid is nearest to that value. These two steps are alternated until a termination criterion is met,

We can choose the k values that demonstrate the clusters in various ways. 'For loop' is the main engine of the method, wherever we estimate each value for k chosen values and set it to nearest cluster, in which 'nearest' describes nearest to the centroid of the cluster. Notice that we can update the centroid of cluster to values set to it. Still, the centroid of cluster trends not to migrate appreciably afterwards only values close the cluster are probably to be set.

Algorithm 1 Outline of the K-means algorithms

Require: Initially choose k points that are likely to be in different clusters
  Make these points the centroids of their clusters;
  for each remaining point p **do**
    find the centroid to which p is nearest
    Set p to the centroid of cluster
    Adjust the centroid of that cluster to account p
  end for

An optional step at the end is to fix the centroids of the clusters and reassign each value, including the k initial values, to the k clusters. Usually, a value p is assigned to the same cluster in which it was placed on the first pass. However, there are cases where the centroid of $p's$ original cluster moves quite far from p after p was placed there, and p is assigned to a different cluster on the second pass [28]. In fact, even some of the initial k values could finish being re-set.

Subsets of S_j are formed after calculating social network measures. Sets are defined as FD, MF, FMF, WF. Values are numerical and the last step in the clustering process takes each value belonging to a set and associates it with the largest centroid. Elements in the area surrounding that centroid compose the item pairs and vectors to input to the item-to-item filtering algorithm.

3.5.2 IMPLEMENTATION OF ITEM-TO-ITEM ALGORITHM

LinkedIn has lots of employees and stores lots of jobs. Our dataset contains limited part of employment information in Turkey. Employment information is the total number of people gainfully employed or working. If worker's sets of chosen jobs have a rich social similarity, we can assume two employees are related. Also, if two jobs chosen by employees with rich social similarity, we can assume they are related. Notice that, when we could think that same profile pages to have similarity of social distance measure above 94%, relation of two employee is too high and it is improbable to have social similarity. Exactly, a similarity of social distance measure like 16% may not be enough to classify employee with same choices. We might reach similar conclusion owns for jobs; connections of user-user or item-item are not required to be so high to be critical.

As we discuss in Chapter 2.3, collaborative filtering(CF) recommendation needs various distance measures, besides determining related employees or jobs. E.g., although only a few of employees will be associated with , two *LinkedIn* employee who choose IT might choose many IT jobs.Nonetheless, we could uncover that IT jobs are respectively related and put them in one set by combining similarity-finding with K-means clustering and social network measures. So, by requesting either they made chooses within many of the same sets, we are able to get a more effective idea of item-similarity.

Collaborative recommendation algorithm recommends jobs to individual workers on a set of jobs that are known to be of interest to the worker, such as a set of jobs previously worked by the worker. In the disclosed representations, the algorithm is used to recommend jobs to workers of a social network. The algorithm produces the recommendations using a earlier-produced table which maps jobs to lists of 'similar' jobs. In the table, the similarities depend on the mutual interests of the group of workers. For instance, in this representation, interaction between chosen jobs by workers are similarity factor.(i.e., jobs A - B are related since wide distribution of the workers who chose job A also chose B). In addition to similarity factor, the table contains values that display rates of similarity between exclusive jobs. The algorithm

produces personalized recommendations in that way : it fetches from the table set of similar jobs complementing to the jobs seen to be of concern of the worker. It properly merge and sort the similar jobs in one set using the similarity values. and filter to produce a sorted set of recommendations. These similar jobs lists are appropriately combined into a single list, which is then sorted (workers on combined similarity scores) and filtered to generate a list of recommended jobs [6].

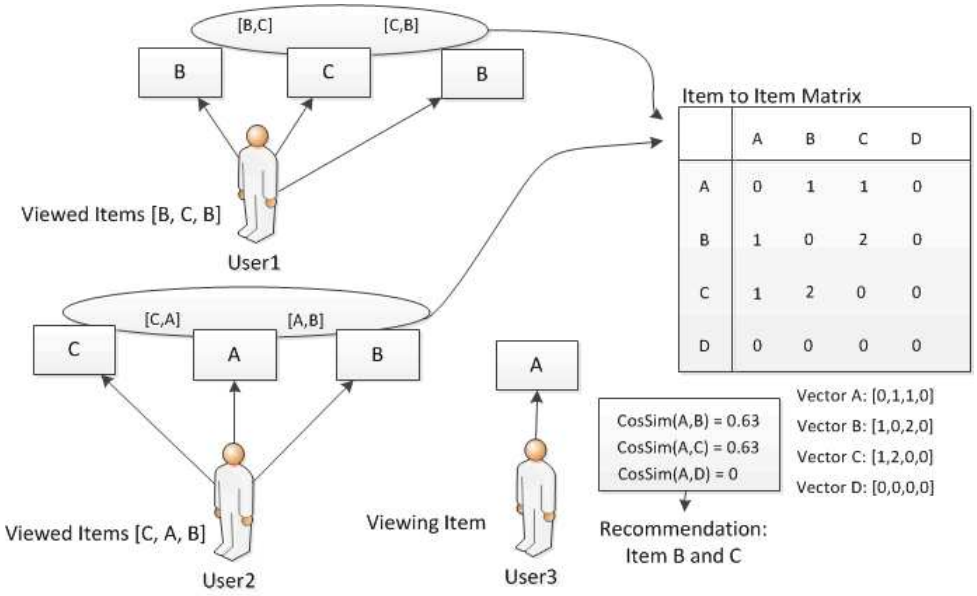


Figure 3.4: Item-to-item recommendation example (simplified illustration).

Hence in the item to item recommendation using of similar jobs is critical and composition of recommendation method part is a upper level perspective of how we are doing it. Firstly, each job is combined with the list of worker who have tried to it. Then, we can compute the distance between two jobs with the normalized measure of the cosine distance(think each item as a vector, with a 1 in the *ith* job. Either worker i has chosen it, and 0 or else). After that we implemented item-based collaborative filtering algorithm using different similarity parameters for measurement calculation between normalized vectors of choices. As we mentioned in Chapter 2.3, The cosine-based and euclidean-based approach defines the similarity between two workers x (worker1) and y (worker2) as :

$$simil(x,y) = cos(\vec{(x)},\vec{(y)}) = \frac{\vec{x}\cdot\vec{y}}{||\vec{x}||_2 * ||\vec{y}||_2} = \frac{\sum_{i\in I_{xy}} r_{x,i} ry,i}{\sqrt{\sum_{i\in I_{xy}} r_{x,i}^2}\sqrt{\sum_{i\in I_{xy}} r_{y,i}^2}} \tag{3.6}$$

$$simil(x,y) = ||x,y|| = \sqrt{\sum_{i=1}^{n} |x_i - y_j|^2} = \sqrt{\sum_{i=1}^{n} r_{x_i-y_j}^2}\sqrt{\sum_{i=1}^{n} r_{x_j-y_i}^2} \tag{3.7}$$

There are two other ways to calculate similarity between jobs' matrix in Chapter 2.3. These are Manhattan Distance based similarity and Minkowski Distance-based similarity.

3.5.3 COMPOSITION OF RECOMMENDATION METHOD

Formally, the recommendation problem can be formulated as follows. Let U be the set of users and I be the set of all possible items. Let also r be a utility function that measures the usefulness of item i to user u, i.e., $R : UxI \rightarrow V$, where V is a totally ordered set (e.g., non-negative integer values or real values within a given range). Then, for each $u \in U$, the recommendation problem consists in finding the item i^* that maximizes the utility of u, i.e.,

$$i^* = \underset{i \in I}{\text{argmax}} R(u, i) \tag{3.8}$$

In most cases, the utility of an item is represented as a rating, that is, an integer value that indicates how much a particular user liked or disliked that particular item.

Algorithm 2 General recommendation algorithm

Require: *a* user u
  OUT: an item i^* to be recommended
  for all unrated items i of user u, **do**
    compute a prediction $R(u, i)$ using some algorithm.
    Recommend the item i^* with the highest prediction.
  end for

However, we propose a compound method that has three steps. We choose the item-to-item collaborative filtering algorithm, which matches each of a user's choices to similar choices [31]. It studies a class of item-based recommendation algorithms to produce predictions for users.

In the first step, the method elaborates social network distances and friendship to help recommender systems customize the target user set. Friends are seen as more qualified to make good and useful recommendations compared to recommender systems primarily because they are assumed to know more about the recommendee.

To find people who are similar to the user, the system divides the worker's friends (target set) into subsets and treats the task as a social clustering problem. Clustering is done on social measures in the second step. Three versions of these measures are common mutual friends, social graph distance, and relationship closeness. Recommendations are not made in rational isolation, which means that they are not evaluated merely by their informational value, rather they are delivered within an informal community of professional network and a social context.

In the second step, the clustering algorithm works on social-related worker sets to build a job table and similar-job pairs for people who tend to do similar work. K-Means is a rather simple but well-known algorithm for grouping objects and clustering. All objects need to be represented as a set of numerical social network measures. The algorithm then randomly chooses k points in that vector space to serve as the initial centroids of the clusters. All objects are then each assigned to the center to which they are closest. Distance measure is chosen by the algorithm and determined by the social features. At the end of the target user selection process, item table (matrix) and pairs are ready for processing from the recommendation algorithm.

Algorithm 3 Outline of Scalable Collaborative Filtering Algorithm and Full Execution Steps

Require: Minimum friend count arg_1 to choose test user
  Initially choose v_x user that are likely to have more than arg_1 friends
  Choose a target(company) that user has chosen(worked) before the current one.
  Get the friends of user to Set of T : $T_{v_x} = \{edge(v_1, v_2).... \in T\}$
  if one of the friends have also chosen(worked) target before **then**
    Calculate the all edges' weight in graph with one of social network measure
    For example : $v_x : n, v_y : k | FD_{n,k} = \sum_{i=0}^{n-k} \binom{n-k}{i}$
    Get the edges' weight scores to list : $ListFD : (FD_{v_1,v_2}, FD_{v_1,v_3}, FD_{v_1,v_4}..., FD_{v_1,v_i})$
    Cluster the social graph with K-means algorithm by using score list.
    for each target(company) in clustered users' , I_1 **do**
      for each user W who worked I_1 **do**
        for each target I_2 chosen by user W **do**
          Record that a user choice I_1 and I_2 and mark the recorded pair on the matrix
          Finalize and normalize the item-to-item matrix to define distance algorithms.
        end for
      end for
    end for
    Choose the distance function. For Example : $d = \cos(\vec{i}, \vec{j}) = \frac{\vec{i} \cdot \vec{j}}{\|\vec{i}\|_2 * \|\vec{j}\|_2}$
    for each target I_2 **do**
      Compute the cosine distance between I_1 and I_2 : d
    end for
    Order and Evaluate the predictions by sorting distance values d
  end if

In third step, highly recommended items are defined by computing a similarity metric, such as cosine distance, on the vectors from the matrix table. In this way, item-to-item recommendation can calculate an ordered prediction for users. A crucial point for the item based collaborative-filtering approach is computation of the relationship between two elements i and j followed by selection of the most similar items $S_{i,j}$. It is possible to compute the similarity between two items in various ways, but a common method is to use the cosine mea-

sure, in which each vector corresponds to a company rather than a worker, and the vector's M dimensions correspond to persons who have worked at that company. Given a similar-company table, the algorithm finds companies similar to those that each person has worked at, aggregates them, and then recommends the most popular or correlated companies.

For evaluation of working prediction, we typically have a dataset consisting of items representing where each person has worked. After selecting a worker for experiment, some of her/his choices are hidden, and the algorithm is asked to recommend a set of items indicating where the worker will work. The algorithm collects the results of the recommendation and accuracy of choices for further ROC curve analysis.

Note that an alternative formulation of the recommendation problem is increasingly being considered. Instead of trying to predict every unknown choice, the recommendation problem can indeed be reduced to the problem of learning to rank. In that algorithm, the goal is to build a model to predict how a particular user would order the items he did not rate, from the one he would tend to the most, to the one he would definitely not tend to.

3.6 EVALUATION

Several key decisions regarding datasets underlie successful evaluation of a recommender system algorithm: Can the evaluation be carried out off-line on an existing dataset or does it require live user tests? A few examples help clarify these decisions.

Evaluations are completed using off-line analysis, a variety of user experimental methods, and a total combination of the two. Much of the work in algorithm evaluation is focused on off-line analysis of predictive accuracy. In such an evaluation, the algorithm is used to predict certain withheld values from a dataset, and the results are analysed using one or more of the metrics. Such evaluations have the advantage of facilitating quick and economical large evaluations, often on different dataset parts and algorithms at once. Once a dataset is available, conducting such an experiment requires us to run the algorithm on the appropriate subset of that data. If the dataset includes timestamps, it is possible to 'replay' a series of choices and recommendations off-line [32]. Each time a choice is made, the algorithm first computes the prediction for that item based on all prior data; then, after evaluating the accuracy of that prediction, the actual choice is entered so that the next item is evaluated [32].

In the following parts, we evaluate our recommendation scheme and recommender algorithm and show its variety and quality of them. Firstly, our proposed methodology which uses different social distance measures and item to item collaborative filtering engine is compared to each other using random parts of the dataset. Workers in these datasets have present and past employment, so we can evaluate the prediction with real job choices. The proposition of this study is actually to try the find an answer to these question: How can we examine our approach(social network measure-based user clustering) concerning quality and variety of recommendations with the item to item engine ?

In the second part, the recommendation algorithm does not estimate the employee's specialities or jobs, like video rankings, yet tries to advise companies for which they may work. To evaluate the working recommendation, we usually test a dataset consisting of items representing where each employee has worked. We formerly chose the testing worker, take few of her/his past choices, and look at the recommendation system to estimate the list of companies that the worker is now going after. Herlocker et al. [33] suggested using receiver operator characteristic (ROC) curves as one measure to evaluate recommender systems. An ROC curve is a curve that shows the hit/miss rates for different classification thresholds. Currently, this research uses parallel(ROC)curves to evaluate the proposed recommendation algorithm. We calculate results using different parameters and compare them via (ROC)curves as a evaluator in this study (i.e., the recommendation algorithm says 'a person will have worked in company' is a predicted positive outcome and 'a person will have not worked in company or have no information about company' is a predicted negative outcome).

3.6.1 SIMULATING USER BEHAVIOUR

We take a so-called backtesting approach to evaluation that is common in recommender systems literature [5]. Whether backtesing had been used in the time of former period, this evolution method tries to classify the efficiency of a approach. Backtesting has historically been performed by big companies and professional banking systems due to the expense of obtaining and using detailed datasets. Nonetheless, recommendation is progressively employed on a larger base and autonomous web based application for backtesting have used. It is devoted to deficiency, despite of the technique is widely used.

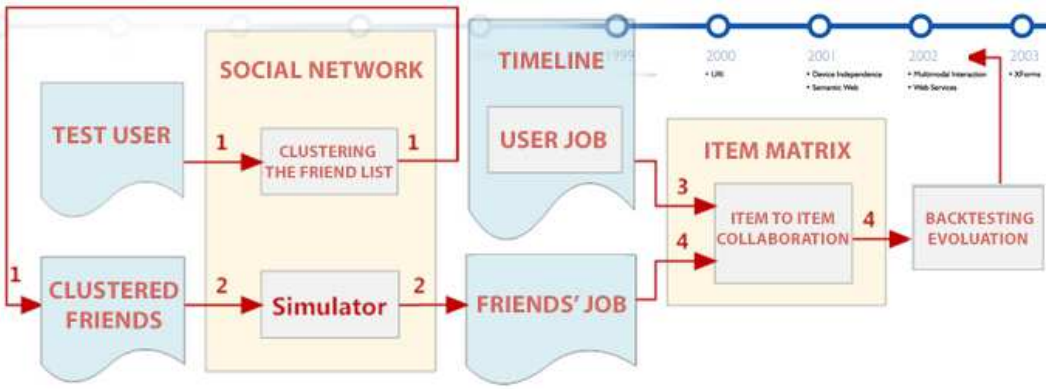


Figure 3.5: The flow diagram of recommender methodology and backtesting steps.Diagram contains simulation of recommendation test and evaluation steps.

We need to correctly simulate the on-line operation to evaluate our recommender off-line. Simulating the on-line steps is a process in which the algorithm produces recommendations or predictions and the worker fixes the advises. To do that, evaluation method stores historical

employment info, and after hides a few of these choices to test this information of which prediction a worker will affect or how a worker will choose a job. It is an important step to choose the hidden choices and we can introduce different ways to do that. But, it is favourable that selection of method be done in a manner that investigates the study domain as closely as possible.

Luckily, if we own timestamps for worker choices, after we are able to pretend which the recommender guesses might have been had it been working simultaneously with the datasets were gathered. We are able to start for any convenient data for calculating guesses, and skip over worker choices in temporary order, trying to guess each collection and after making the collection usable for next guesses. Because of the size of the datasets a simpler approach is to randomly sample test users, randomly sample a time just prior to a user action, hide all selections (of all users) after that instant, and then attempt to recommend items to that user. This protocol requires changing the set of information given prior to each recommendation, which can still be computationally quite expensive.

An even cheaper alternative is to sample a set of test users, then sample a single test time, and hide all items after the sampled test time for each test user. This simulates a situation in which the recommender system is built as of the test time, and then makes recommendations without taking into account any new data that arrives after the test time. A final solution is to ignore time (because the objective is to develop a scalable system for recommendation). We take as an example a group of workers, after chose the number n_a of jobs to conceal for each worker, after chose n_a jobs to conceal. Thus, our approaches presume that this temporary situation of worker selections is trivial. All next different selections divide the dataset to only one test and training group. Choosing a way that is most proper for our concern and research interest is necessary for the given constricts.

Using a constant number of hidden elements or a constant number of known elements for a test user (so called given n or all but n protocols) [32] is used in lots of published papers. Also this protocol has a common usage. It is valuable for analysing the situations in which they work best and identifying the problems in algorithms. Yet, when we try to decide selections on the algorithm used in our method, our question is whether we really take care of showing selections solely for users who have taken particularly n elements, or are predicted to recommend particularly n elements more. Consequently, the results computed using these protocols have biases to n and n number of prediction that only make them reliable for making recommendation with on-line performance of the our method.

3.6.2 EVALUATION METRICS

3.6.2.1 CORRECTNESS OF PREDICTION

Correctness of prediction is the most conferred subject in literature of recommendation systems like the other machine learning sub-areas. Lots of recommendation algorithms and

systems contains a prediction implementation. The implementation of prediction might foresee user choices over elements (i.e., ranking of videos) or the possibility of purchasing (i.e., shopping).

A simple expectation from the recommendation algorithm is giving more correct predictions to consumer. So, the main aim in the research area is revealing algorithms that give better predictions.

A basic assumption in a recommender system is that a system that provides more accurate predictions will be preferred by the user. Thus, researchers set out to find algorithms that provide better guesses.

Correctness of prediction is mostly separated from the user profile, and so is calculated in an off-line test. Checking correctness of prediction in a user aspect calculates the correctness given a recommendation. This approach is closer to the true correctness for the on-line application and is a different approach from the guess of user behaviour without giving any recommendation.

We employ two famous formulas to calculate the correctness and the prediction quality of our propositioned method (MAE - Mean Absolute Error, RMSE - Root Mean Square Error)[50] and compare it with other recommendation algorithm (Chapter 4).

The MAE formulas is described and detailed as follows:

$$MAE = \frac{1}{|T|} \sum_{(u,i \in T)} |\hat{r}_{ui} - r_{ui}| \tag{3.9}$$

Root Mean Squared Error (RMSE) is the most popular metric used in evaluating accuracy of predicted ratings. The system generates predicted ratings $\hat{r}_{ui}$ for a test set T of user-item pairs (u, i) for which the true ratings $\hat{r}_{ui}$ are known. Typically, $\hat{r}_{ui}$ are known because they are hidden in an off-line experiment, or because they were obtained through a user study or an on-line experiment. The RMSE between the predicted and actual ratings is given by

$$RMSE = \sqrt{\frac{1}{|T|} \sum_{(u,i \in T)} (\hat{r}_{ui} - r_{ui})^2} \tag{3.10}$$

We use average Root Mean Square Error(RMSE) in our unadjusted test dataset. E.g., average Root Mean Square Error(RMSE) gathered from test dataset could be densely affected from the error on a hardly any dense jobs, if the test dataset contains an unadjusted distribution of jobs. Hence, it is preferred to calculate Root Mean Square Error(RMSE) differently for each job and then take an average of all jobs, if we require a calculation that is typical example of the prediction error on any job. Equivalently, if the test dataset contains an unadjusted worker distribution and we want to learn the prediction error a random picked worker could encounter, we could measure a per worker average Root Mean Square Error(RMSE).

We explain here one broad class of prediction accuracy measures that measures the accuracy of usage predictions and apply them:

3.6.2.2 MEASURING USAGE PREDICTION

For each test worker we randomly choose a job from their jobs, and generate recommendations by using the friends as collaboration material. If a worker's selected jobs are predicted at the top of the ranked result list, i.e., if the algorithm is able to correctly predict the worker's next job in those selected jobs, then the algorithm is considered to perform well. In the rest of this thesis, we will refer to the worker we are trying to recommend jobs for as the active test worker and his friends' jobs as that worker's active jobs.

This approach of withholding test jobs from worker friends also serves as an extra justification for removing the duplicate jobs. When a duplicate job is selected for an simulated worker, then it is not present in the friends list anymore. If the recommendation algorithm does not know about the job from its presence in the friends' set, it can never be recommended to the active worker. This puts an artificial ceiling on possible performance and makes for an unfair evaluation of the algorithms. If any job occurs at least twice in the entire data set, then the odds of that job being selected for evaluation in both cases are much smaller, sketching a more realistic picture of performance. Instinctively, the odds of jobs not being recommendable decrease as the job filtering threshold increases even further, the trade-off here being between impractical evaluation.

In our evaluation, we adopt an Machine Learning perspective by treating each of the workers as a separate process. The selected jobs for each worker make up the related jobs for which we have relevant experience. For each worker a ranked list of jobs is produced and evaluated on where these selected jobs show up in the result list. While it is certainly possible and very likely that the recommendation lists contain recommendations that the worker would find related or interesting, we cannot know this without the worker experiencing them. This means that because our relevance experience correspond to jobs added to a worker's job list, we can never have any jobs judged as being not related without worker interference.

Table3.4: Categorization of the feasible conclusions of recommendation of a job to the worker

	Recommended	Not recommended
Worked	True-Positive (tp)	False-Negative (fn)
Not Worked	False-Positive (fp)	True-Negative (tn)

In evaluation of working prediction, we typically have a dataset consisting of items indicating where each person has worked. After selecting a worker for experiment, some of her/his choices are hidden, and the algorithm is asked to recommend a set of items indicating where

the worker will work. After that, we get four potential response for the recommended jobs, as presented at Table 3.4. Also, we can get these responses for hidden jobs.

In this scenario, we are enforced to estimate that un-worked jobs could not have been chosen although they had been recommended, because of the data elements are not mostly gathered processing the recommendation systems in evaluation part. This assumption may be false; for example, a user may not have worked at that job because she was unaware of its existence, but after the recommendation exposed that item the user can decide to select it. In this case, the number of false positives is over-estimated [32]. We can count the number of examples that fall into each cell in the table and compute the following quantities:

$$precision = \frac{\#tp}{\#tp + \#fp} \tag{3.11}$$

$$truepositiverate(tpr) = \frac{\#tp}{\#tp + \#fn} \tag{3.12}$$

$$falsepositiverate(fpr) = \frac{\#fp}{\#fp + \#tn} \tag{3.13}$$

In this approach, the presented recommendations is predetermined and it is favourable that algorithm is evaluated over lengths of recommendation list, instead of a fixed length. Hence, curves can be calculated by measuring true positive rate(tpr) to false positive rate(fpr) or precision(p) to recall(r). We formerly present the curves as Receiver Operating Characteristic(ROC) curves. Precision recall curves indicate the proposed of suggested jobs that are chosen while ROC curves indicate the proposed of jobs that are not chosen that end up being suggested, during the time both curves measure the proposed of chosen jobs that are literally suggested.

Measure of precision(p) explains the proposed recommendations that are really appropriate for the worker. Either the inappropriate recommendations show a little or more fraction of the inappropriate companies that might have been suggested, the false positive(fp) rate might not be as proper as proposed of the relevant jobs the algorithm advises to the worker, hence a curve of *tpr-fpr* is appropriate for evolution tests.

Given three social network measures, we can compute ROC(curves) for each choice. Either a ROC(curve) totally overtops the other(ROC), the suggestion as to which algortihm is superior is simple. Hence, when the ROC(curves) overlap, the suggestion is fewer apparent. Maybe, the suggestion relies on our method in advance. Information of the application dictates which region of the curve the decision is based on. Measures that summarize the precision recall of ROC curves, such as F-measure [34] and the Area Under the Curve (AUC) [35], are useful for comparing choices independently of our approach.

When using normalized units, the AUC is equal to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one (assuming 'positive' ranks higher than 'negative'). It can be shown that the area under the ROC curve is

closely related to the Mann-Whitney U. [36], which tests whether positives are ranked higher than negatives. It is also equivalent to the Wilcoxon test of ranks [36]. The AUC is related to the Gini coefficient G_1 by the formula :

$$G_1 = 2AUC - 1 \tag{3.14}$$

$$where : G_1 = 1 - \sum_{k=1}^{n} (X_k - X_{k-1})(Y_k + Y_{k-1}) \tag{3.15}$$

In this way, it is possible to calculate the AUC by using an average of a number of trivial approximations.

The machine learning community most often uses the ROC AUC statistic for model comparison. However, this practice has recently been questioned on the basis of new machine learning research that shows that the AUC is quite noisy as a classification measure and has some other significant problems in model comparison [37]. A reliable and valid AUC estimate can be interpreted as the probability that the classifier will assign a higher score to a randomly chosen positive example than to a randomly chosen negative example. However, critical research [37] suggests frequent failures in obtaining reliable and valid AUC estimates. Consequently, the practical value of the AUC measure has been called into question [38], raising the possibility that the AUC may actually introduce more uncertainty into machine learning classification accuracy comparisons than resolution. One recent explanation of the problem with ROC AUC is that reducing the ROC Curve to a single number ignores the fact that it is about the trade-offs between the different systems or performance points plotted and not the performance of an individual system, as well as ignoring the possibility of concavity repair [38].

3.6.3 EXPERIMENTAL RESULTS

In this part, we show our test result and success gained by implementing item based recommender methods to generate recommendations. Results are separated to two parts: performance and quality. For quality of predictions, initially parameter sensitivity is determined before processing the main test. These parameters contain the count of K-Means root, the impacts of various distance measures, similarity measures and rank of the friendship ratio x. To determine the sensitivity of different parameters, we focus on the off-line evaluation result and further separate it into social-based portioning, which we use to test the parameters. First of all, we outline our motivating outcomes, after which we discuss the impacts of different parameters and clustering methods.

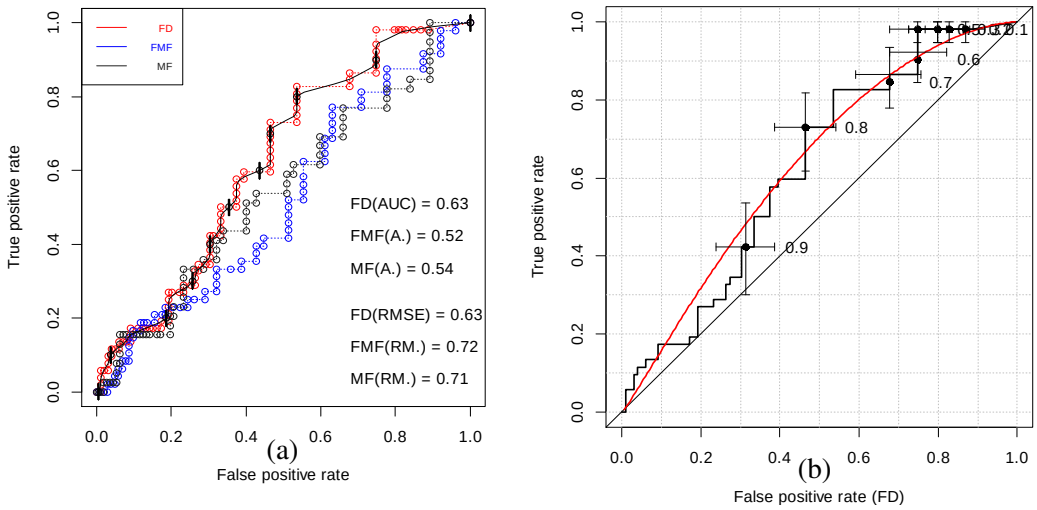


Figure 3.6: (a)Comparison of the ROC Curves (with inclined form), AUC, and RMSE predicted by our models against social measures distances.(b) Average and cut-off points of three ROC Curves produced by evolution algorithm.

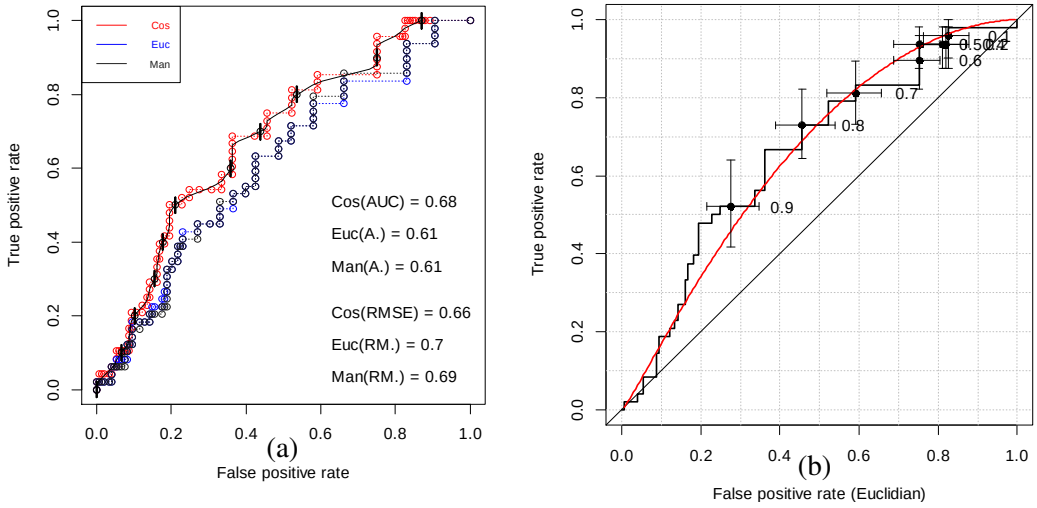


Figure 3.7: (a) Comparison of the ROC Curves (with inclined form), AUC, and RMSE predicted by our models against vector distance functions.(b) Average and cut-off points of three ROC Curves produced by evolution algorithm.

First, we test the social network measures' clustered results with cosine similarity against

the others. We then compare similarity results using friend's distance clustered subset. A presumption of binary relevance is generated from (ROC)curves : Recommended jobs are classified successful(relevant) or unsuccessful(non-relevant) and that can be assigned 1 and 0. As a result of this presumption the ordering among related jobs has no concern on the (ROC)curve metric. When all related jobs occur before whole non-related jobs in the recommendation list, we can get a good (ROC)curve results.

We mark the hit/miss rates on x - y coordinates, mutually. Different thresholds build a serial of points and the shape of serial is like a curve. In the literature, the area under curve(AUC) is a performance measure, with perfect performance indicated by an area of one and random guessing indicated by an area of 0.5 [38]. Figure 3.6 compares the size of this area to a perfect curve to determine the success of clustering and recommendation. Standard distribution of curves relative to each other (red curve) proves that the pre-ordered prediction increases the regularity of the hit rate over unordered for recommendation.

Table3.5: Overall accuracy of distance measures and network measures

	Hit Rate	Miss Rate	Non Rec.	AUC
FD-Euc	%38.62	%23.16	%29.22	*0.634*
FMF-Euc	%24.28	%36.02	%40.72	*0.523*
MF-Euc	%26.14	%29.53	%45.33	*0.546*
FD-**Cos**	%38.62	%23.16	%29.22	*0.617*
FD-**Euc**	%37.62	%24.16	%29.22	*0.634*
FD-**Man**	%37.02	%24.66	%29.22	*0.619*

Table 3.6 and Figure 3.7 show three friend clustering hit rate results: Test on Friend Distance (FD) measure, Test on Mutual Friend (MF) measure, and Test on Friend's Mutual Friends (FMF) measure. All accuracy measures were collected over 10 randomly cross-validated runs. The **FD** output with Euclidean-based Similarity achieves a significant boost over the next highest.

Table 3.6 also shows the results for three similarity measures used in the item-to-item filter algorithm hit rate results: Test on Cosine Distance (Cos), Test on Euclidean Distance (Euc), and Test on Manhattan Distance (Man). All accuracy measures were collected over the same schema and after clustering all methods, the **Euc** output achieved more boost over the next highest.

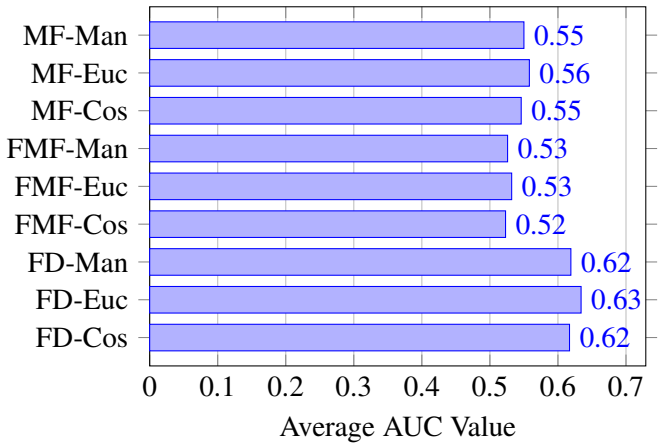


Figure 3.8: Comparison of AUC values prediction on all evaluation tests

The experimental results of applying hybrid collaborative filtering techniques to generate predictions are presented here. Results are separated to two parts: performance and quality. For quality of predictions, the type of clustering measure is determined before processing the main test. In item-to-item algorithm, the type of distance measures is tested at each step. Different types of determination change the performance of the hybrid approach.

3.6.3.1 IMPACT OF SOCIAL MEASURES

We implemented the four different social network distance measure algorithms and correlations explained in (Section 2.5). We also tested our approaches on the dataset. For each social network distance measure, we carried out the algorithms to calculate the neighbourhood. We tested our item-to-item collaborative filter recommender to produce the recommendation. To calculate Root Mean Square Error(RMSE) these experiments carried out three different subdataset. These are small subset of the total data and generated for experimental evaluation. Figure 3.9 shows the test results obtained. It can be seen from the results that compensating the worker-average for cosine-similarity-computation has pure advantages, as the RMSE is notably fewer in this graphs. So, we choose the cosine similarity measure for the rest of our experiments to investigate other parameters.

The social network distance measure algorithm has a significant impact on the prediction quality. To determine the actual impact, we performed an experiment in which we varied the number of tests conducted on the different parameters to be used and computed RMSE and AUC.

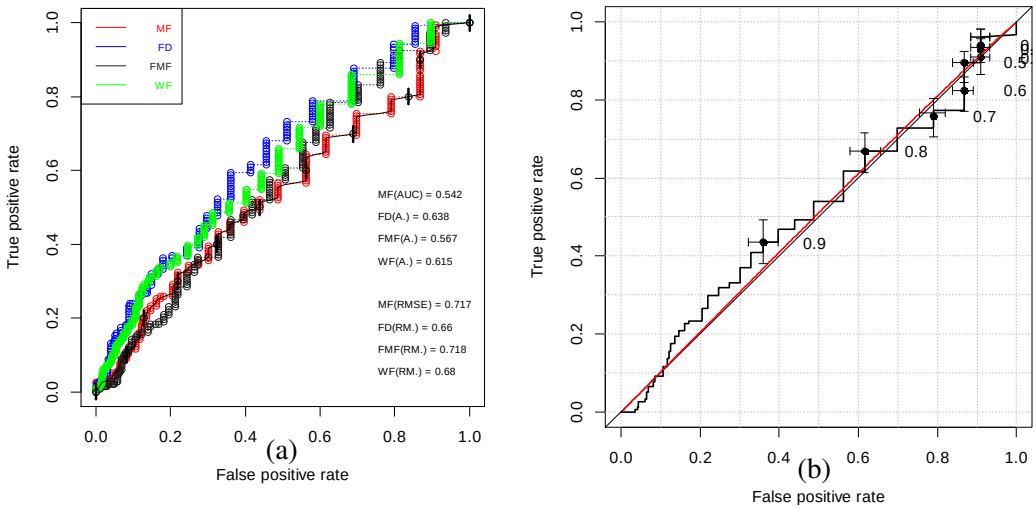


Figure 3.9: (a) Comparison of the ROC Curves (with inclined form), AUC, and RMSE predicted by our models against social measures distances. Over 1000 users tested with random friendship sizes.(b) Average and cut-off points of four ROC Curves produced by evolution algorithm.

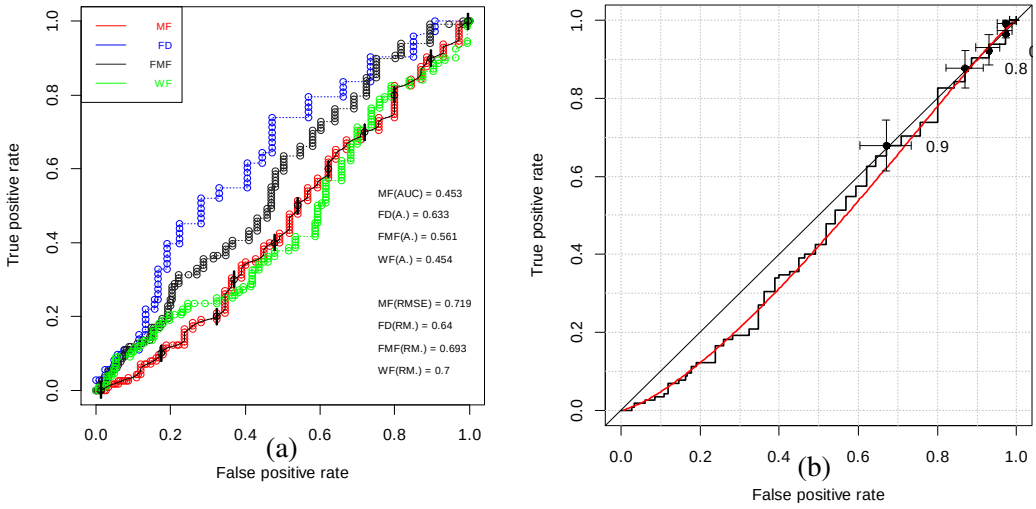


Figure 3.10: (a) Comparison of the ROC Curves (with inclined form), AUC, and RMSE predicted by our models against social measures distances. Over 300 users were tested, with minimum 20 friendships sizes.(b) Average and cut-off points of four ROC Curves produced by evolution algorithm.

The tests shown were conducted over different parts of the dataset and with different parameters. First, we tested our algorithm with 1000 users without considering the size of the friendship. We then got a more specific user group in order to compare the results. All 300 users had more than 20 friends in their social networks.

Table3.6: Accuracy values of social network measures algorithms in graphs 3.9 and 3.10

	Hit Rate	Miss Rate	Non Rec.	AUC	RMSE
FD-1	%29.62	%27.6	%43	*0.638*	*0.660*
FMF-1	%19.28	%36.02	%48.4	*0.567*	*0.718*
MF-1	%20.31	%30.2	%39.33	*0.542*	*0.717*
WF-1	%18.24	%20.53	%53.33	*0.615*	*0.680*
FD-2	%39.12	%59.16	%0.22	*0.633*	*0.640*
FMF-2	%37.74	%59.15	%2.23	*0.561*	*0.693*
MF-2	%30.14	%62.53	%4.88	*0.453*	*0.719*
WF-2	%26.14	%59.53	%37.19	*0.454*	*0.700*

The results obtained are shown in Figure 3.10. It is clear that social network distance does affect the quality of prediction. However, the two methods show different types of sensitivities. The FD algorithm improves the result with a more calculating cycle, while FMF shows a decrease in prediction quality with a less calculating cycle. Results prove that proposed algorithm do better concerning variety, quality and has not a loss in success, despite it is more scalable. Also note that, the diversity in success decreases as the number of data size increases.

3.6.3.2 IMPACT OF PARAMETERS

The wideness of social network has a powerful effect on the correctness and quality of prediction. we did a test in which we evolved the number of friends used and calculated RMSE to analyse the sensitivity of this parameter. Gathered results are presented in Table 3.8.

It is clear that the amount of the friendships size does affect the recommendation quality. However, the two parameters present different types of sensitivities. The algorithm improves as we change the minimum the amount of the friendship size from 0 to 29. The rate of grow the subsequently lessens and the ROC(curve) tends to be flat. Conversely, the social network algorithm shows a decrease in execution performance with increased friendship amount. Considering both trends, we selected 20 as our minimum choice for friendship amount.

On obtaining the optimal values for the parameters, we compared both of our vector distance functions with the item-to-item algorithm. The results obtained are depicted in Figures 3.11 and 3.12. It can be seen from the charts that the cosine distance outperforms the others at all

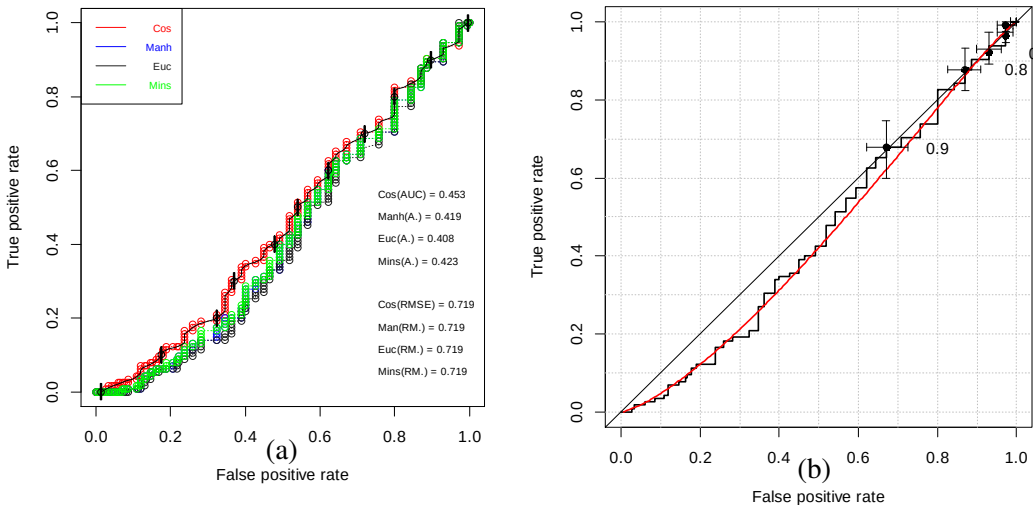


Figure 3.11: (a) Comparison of the ROC Curves (with inclined form), AUC, and RMSE predicted by MF social measure model against vector distance functions. Over 300 users tested with minimum 20 friendship sizes.(b) Average and cut-off points of four ROC Curves produced by evolution algorithm.

values of x (neighborhood size, min 20). For example, in Figure 3.11, test scheme has an AUC of 0.453 and all the others show AUCs near 0.4. Similarly, other test schemes with larger user spaces show differences of AUC 0.30, respectively. Not all the algorithms, however, show interesting behavior, generally, vector distance measures do not have an important effect on prediction performance.

Table3.7: Accuracy values of vector distance functions in graphs 3.11 and 3.12

	Hit Rate	Miss Rate	Non Rec.	AUC	RMSE
Cos-1	%33.14	%64.82	%0.02	*0.453*	*0.719*
Euc-1	%33.14	%64.82	%0.02	*0.419*	*0.719*
Manh-1	%33.14	%64.82	%0.02	*0.542*	*0.719*
Mins-1	%33.14	%64.82	%0.02	*0.423*	*0.719*
Cos-2	%26.32	%73.64	%0.22	*0.615*	*0.676*
Euc-2	%26.32	%73.64	%0.22	*0.601*	*0.676*
Manh-2	%26.32	%73.64	%0.22	*0.602*	*0.676*
Mins-2	%26.32	%73.64	%0.22	*0.603*	*0.676*

To determine the sensitivity of the user density of the dataset, we conducted an experiment in which we varied the size from 175 to 1000 in increments of 20%. To obtain each of these

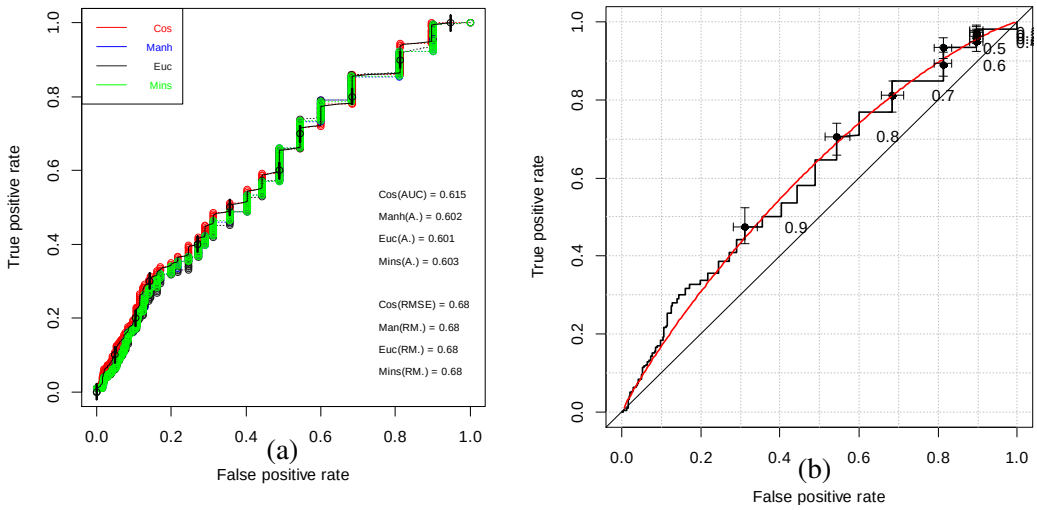


Figure 3.12: (a) Comparison of the ROC Curves (with inclined form), AUC, and RMSE predicted by FD social measure model against vector distance functions. Over 1000 users tested with random friendship sizes.(b) Average and cut-off points of four ROC Curves produced by evolution algorithm.

hit/miss rate values, we operated our algorithm on test dataset using the **MF** and **FD** approach. The mutual results obtained are shown in Figures 3.6, 3.7, and Figures 3.11 and 3.12. It can be seen that the quality of the prediction increases as only us can increase the size of the friendship. The **FD**-**Cos** combination shows better results than the other combination for all user densities.

3.7 RUN-TIME PERFORMANCE AND APPLICABILITY

The experimental results obtained by applying the item-to-item collaborative filtering technique to generate predictions are presented in this section. Results are separated to two parts: performance and quality. For quality of predictions, the type of clustering measure and parameter selection is determined before processing the main test. In the item-to-item algorithm, the type of distance measures was tested at each step. The performance of our approach varies according to the type of determination.

After showing that the item based algorithm provides good quality prediction, we try to compound a scalable algorithm. As explained before, item based collaborative filtering is more static than other approaches. It supports us to pre-compute the job neighbourhood. The preclustering of the friends has lots of improvement points to get more performance. To make the recommender even more fast and scalable, we focused on the impact of friend sizes and the sensitivity of distance parameter on the CPU response time.

Table 3.9 shows the time measures of the total algorithm for three friend clustering hit rate results and item-to-item filtering on the CPU: Test on Friend Distance Measure (FD), Test on Mutual Friend Measure (MF), and Test on Friend's Mutual Friends Measure (FMF). Results proves that proposed algorithm do better concerning of variety, quality and has not a loss in success, despite it is more scalable. Also note that, the diversity in time does not change when the number of friends increases. This point is key to scalability.

Table3.8: Overall performance values and their sizes in the social area

	Avg. Execution Time	Avg. Friends Count
FD-Cos	7.01 s	160.2
FMF-Cos	2.69 s	140.8
MF-Cos	1.53 s	155.2
FD-Euc	7.11 s	160.2
FMF-Euc	2.64 s	140.8
MF-Euc	1.51 s	155.2

As mentioned before, item-to-item similarity is more static than other approaches. It supports us to pre-compute the job neighbourhood. The pre-clustering of the friends has lots of improvement points to get more performance. Figure 3.13 illustrates that all off-line test combinations consume similar CPU time. Our test environment and recommendation methodology are reasonably scaled for improvement.

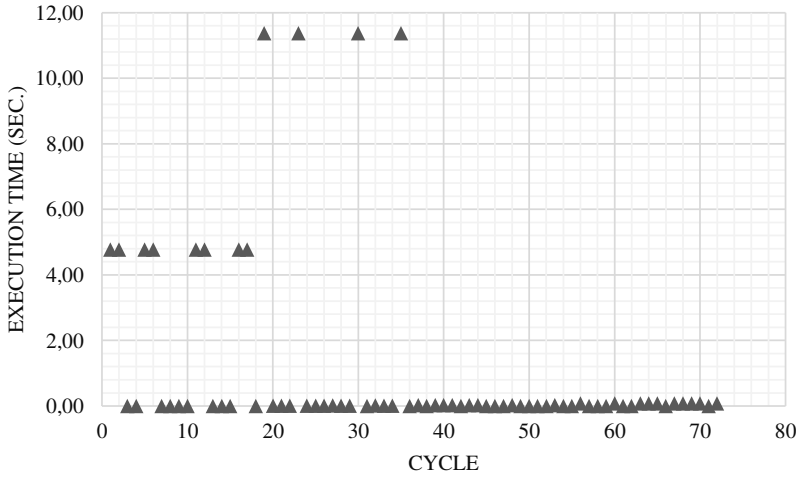


Figure 3.13: Distribution of execution time for each combination of distance measures. The experimental algorithms have 72 execution cycles in total.

3.8 DEMONSTRATION OF IMPLEMENTATION AND EVALUATION

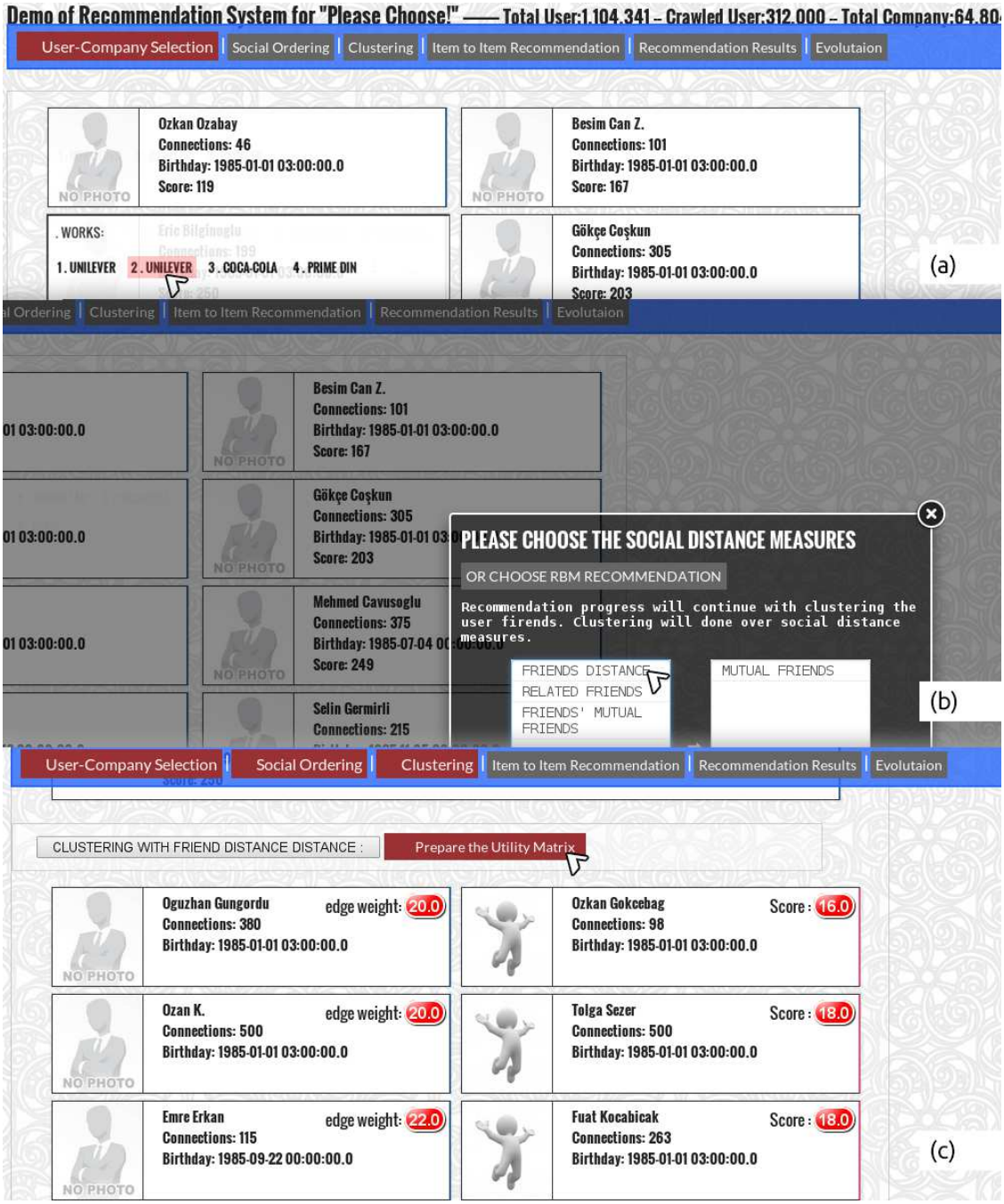


Figure 3.14: Screenshot(a) shows job choice for the first step of backtesting; (b) presents parameter selection of social distance measure; (c) indicates the result of social clustering.

In the figure 3.14, our proposed method elaborates on social network distances and friendship information to help recommender systems customize the target user set. Friends are seen as

more qualified to make good and useful recommendations compared to recommender systems primarily because they are assumed to know more about the recommendee.

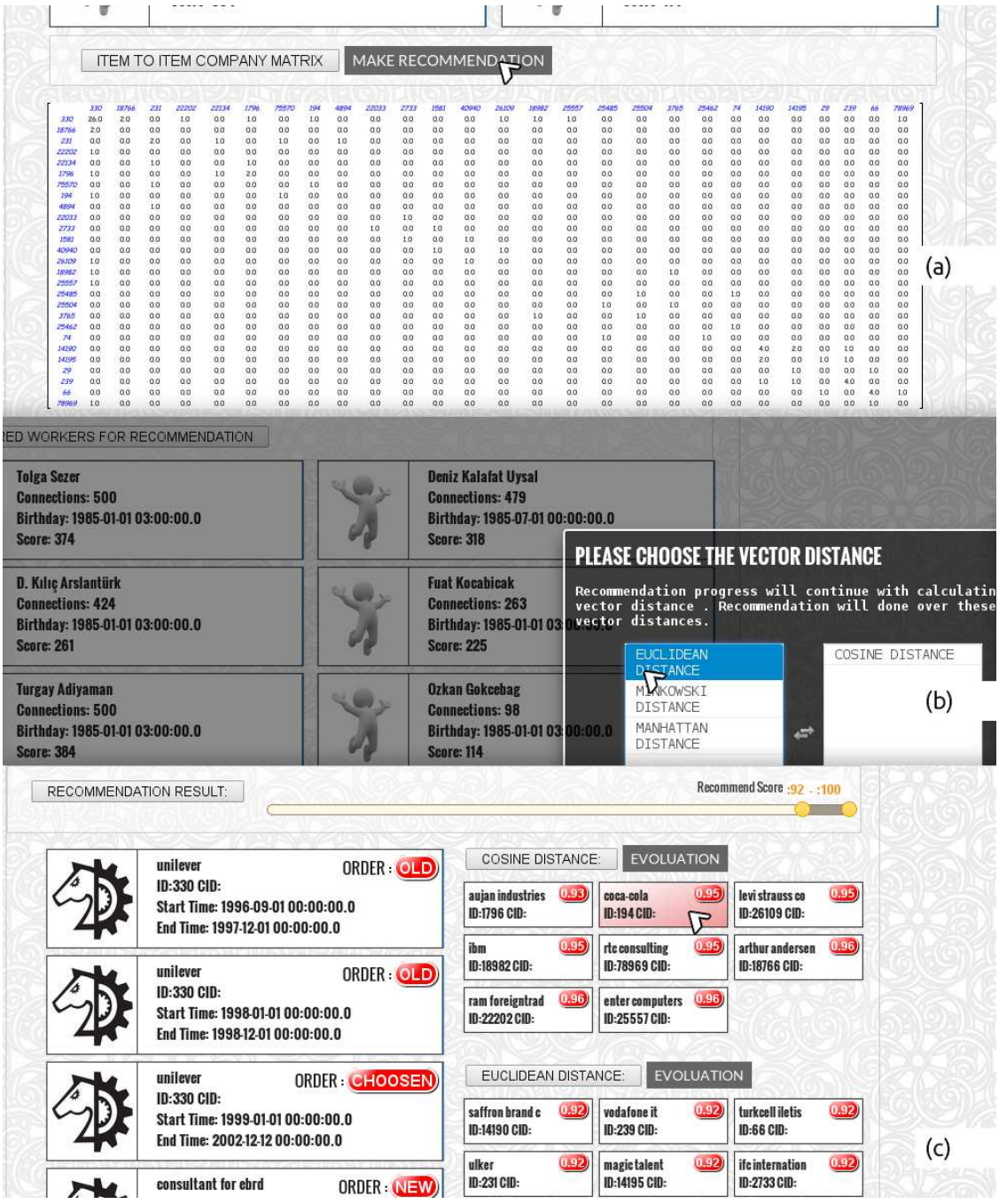


Figure 3.15: Screenshot(a) shows the item-to-item matrix constructed with previous worker choices; (b) presents parameter selection of vector similarity functions; (c) displays the ordered job recommendations and its evolution metrics.

In the screenshot(a), the test algorithm chose a previous job in her/his working timeline and

the prediction is computed from the past employment of them. To find people who are similar to a user, the system divides the users' friends (target set) into subsets, and treats the task as a social clustering problem. The proposed system selects a social network measure and clustering is then done on the social measures in the second step. Screenshot(b) indicates the demonstration of these processes.

The clustering algorithm works on a socially worker set to build a job table and similar job pairs people who tend to do similar work. K-means is a rather simple, but well-known, algorithm for grouping objects and clustering. Screenshot(c) shows the clustered friends of the worker and all objects are represented as a set of numerical social network measures. The algorithm then randomly chooses k points in vector space to serve as the initial centers of the clusters. Subsequently, each object is assigned to the center to which it is the closest. Distance measure is chosen by the algorithms and determined by the social features.

In the figure 3.15, highly recommended items are defined by computing a similarity metric, such as cosine distance, on the vectors in the matrix table (Screenshot(a)). In this way, item-to-item recommendation can compute an ordered prediction for users. A crucial point for the item based collaborative-filtering approach is computation of the relationship between two elements i and j followed by selection of the most similar items $S_{i,j}$.

It is possible to compute the similarity between two items in a number of ways; we can easily see that selection in screenshot(b) however, the most common method is to use the cosine measure, in which each vector corresponds to a company rather than a worker, and the vector's M dimensions correspond to persons who have worked at that company. Given a similar-company table, the algorithm finds similar companies for which the target people have worked, aggregates those companies, and then recommends the most popular or correlated companies in screenshot(c). We are able to compare recall to precision, or false positive rate to true positive rate and backtesting results. The curves of the previous version are identified commonly as the curves of precision-recall, while those of the next version are identified as ROC (Receiver Operating Characteristic curves). Information related to present and past employment data allows us to draw ROC curves and calculate RMSE values after recommendation. While top-N limits the predictions to some predefined number, ordered search results allow the worker to continue to look at jobs highly likely to be of interest to them. This feature allows workers to have query returns sorted by the predicted likelihood that the worker will chose the job. Once again, this helps convert workers into employees

3.9 DISCUSSION

All experimental methods for evaluating and comparing recommendation algorithms have their benefits and their drawbacks. As target of a recommender technique is to entice workers to view, work, like, or otherwise use elements that are new to them and of potential interest, more direct and appropriate way to evaluate an algorithm is to run it in live service and measure the approval rate by real workers. This is generally regarded as one side of the evaluation

spectrum. However, in a scenario where many various algorithms need to be compared, each with their own parameter settings, it is not always practical to explore all these varieties. Each variety would have to be run long enough and with enough workers to be able to draw statistically significant conclusions about the differences in achievement. Additionally, testing poorly performing varieties could result in unacceptable depravity of the live achievement of the techniques for a subset of the worker population [58].

Alternative end of the evaluation spectrum, backtesting and other off-line evaluation methods make it easy to fast prototype and compare various algorithms, without having to rely on or wait for a sufficiently large group of workers to work with the technique and react to the recommendations. As it is still imprecise which recommendation algorithms will perform best in professional social network, we firstly need to whittle down the space of possible varieties before a live comparison with real workers should be attempted. Hence, we focus on backtesting evaluation in this thesis. There are a number of issues with our technique-based evaluation procedure that should be stated to put our results in the correct perspective.

One weakness of off-line analyses such as our technique-based evaluation is that we are limited to an objective evaluation of our results, and based on this we cannot determine whether workers will prefer one particular algorithm over another algorithm [59]. System approval is dependent not just on prediction quality, but also on other more subjective criteria such as the worker interface. A second problem of off-line evaluation is that we only possess unary relevance experiences, which do not allow us to make any assumptions about whether or not unseen elements would be appreciated by the worker. In addition, because backtesting is done on a snapshot of the professional social network's database, it hard to simulate the dynamic nature of live recommendation, where new workers and elements are added continuously, with new trends and old trends coming and going. Such aspects are harder to capture in an off-line analysis.

CHAPTER 4

COMPARISON OF RECOMMENDER SYSTEMS USING RESTRICTED BOLTZMANN MACHINES

As pointed out in Chapters 2 and 3, our methodology and its' algorithms have been proposed to make recommendations. From now on though, and for the rest of this text, focus will be totally on a specific algorithm called restricted Boltzmann machines (RBMs) and comparison of our methodology. The importance of testing this particular model is actually two-induced. First, it was one of the best single model that has been used during the *Netflix* challenge. Every leading team included several variations of this model in their final blending. Second, its applications are not limited to recommender systems. They have been used for various other tasks, such as digit recognition, document retrieval or image de-noising.[39][40][41].

Our approaches to collaborative filtering can handle very large datasets with social connections. In this section of the thesis, we show how a class of two-layer undirected graphical models, called Restricted Boltzmann Machines (RBMs), can also be used to model tabular data, such as a user's choice of job. We present efficient learning and inference procedures for this class of models and demonstrate that RBMs can be effectively applied to our dataset, which contains over 1.4 million worker/job choices, for comparison.

Section 4.1 first reviews the model from which RBMs are derived. Section 4.2 and 4.3 then focuses on RBMs themselves by introducing the complete learning algorithm and then presenting an insightful application example. RBMs are then examined in the context of collaborative filtering in section 4.5. A variation of the model, in which implicit feedback is taken into account, is also examined in this section. Finally, section 4.6 presents a thorough experimental study of the model and of its parameters when tested over *LinkedIn* data. The differences in each approach of the model are also examined.

4.1 BOLTZMANN MACHINES

The Boltzmann machines are a type of neural network invented by David Ackley, Geoffrey Hinton and Terrence Sejnowski [42]. Intuitively, the purpose of these networks is to model the statistical behaviour of some part of our world. What this means is that a Boltzmann

machine can be shown some distribution of patterns that comes from the real world and then infers an internal model that is capable of generating that same distribution of patterns on its own [43]. Typical applications of such a model include pattern classification, generation of plausible patterns, in case we need some more, or reconstruction of partial patterns.

Technically, a Boltzmann machine is a recurrent neural network composed of stochastic binary units with symmetric connections. The nodes in a Boltzmann machine are usually divided into a set of *visible units* that can have data clamped onto them, and a set of *hidden units* that act as latent variables [45]. Units are connected to each other with symmetric connections in any arbitrary way, except with themselves (c.f., Figure 4.1). Each unit i has a binary state s_i and turns either on or off (i.e., $s_i = 1$ or $s_i = 0$) with a probability that is a logistical function of the inputs it receives from the other units j it is connected to:

$$p(s_i = 1) = \frac{1}{1 + \exp(-b_i + \sum_j s_j w_{ij})} \tag{4.1}$$

where b_i is the bias term of unit i and w_{ij} is the weight of the symmetric connection between unit i and unit j. The weights and biases of a Boltzmann machine define an *energy function* over global configurations (i.e., binary state vectors) of the network. The energy of a configuration (v, h), where v is a binary state vector of the visible units and h a binary state vector of the hidden units, is defined as:

$$E(v, h) = -\sum_i b_i s_i^{(v,h)} - \sum_{i<j} s_i^{(v,h)} s_j^{(v,h)} w_{ij} \tag{4.2}$$

If units are chosen at random and continuously updated using equation 4.1, it can be shown that the network will eventually reach a stationary probability distribution (or *equilibrium*) [45] in which the probability of finding the network in any global configuration (v, h) is determined by the energy of that configuration relative to the energies of all other possible configurations:

$$p(v, h) = \frac{\exp(-E(v, h))}{\sum_{u,g} \exp(-E(v, g))} \tag{4.3}$$

Considering equation 4.3, a Boltzmann machine can be viewed as a generative model that assigns a probability to each possible binary state vector over its visible units. Indeed, because of the stochastic behaviour of the units, the network will wander through a variety of states and will therefore generate a probability distribution over all the 2^N possible visible vectors (where N is the number of visible units) [49]. Equation 4.3 determines their respective probabilities. In that context, we want a Boltzmann machine to build an internal model capable of generating over its visible units a particular distribution of patterns (the data), learning amounts to finding weights and biases that define a probability distribution in which those patterns have a high probability, hence a low energy.

Let $P^+(V)$ be the distribution of patterns we want to model and $P^-(V)$ the distribution generated over the visible units of a Boltzmann machine when the network runs freely at equilibrium. Considering the measure to evaluate the distance between the two distributions, learning amounts to minimize:

$$G = \sum_v P^+(V) \ln(\frac{P^+(V)}{P^-(V)}) \tag{4.4}$$

Since equation 4.4 is an indirect function of the weights and biases of the Boltzmann machine, the model can be improved by modifying the w_{ij} 's and the b_i's so as to reduce G. Hence, a simple gradient descent strategy can be used to minimize G. Surprisingly, it can be shown [42] that the partial derivative of G with respect to w_{ij} is as simple as

$$\frac{\partial G}{\partial w_{ij}} = -(< -s_i s_j >^+ - < -s_i s_j >^-) \tag{4.5}$$

where $< -s_i s_j >^+$ is the averaged probability, when data vectors from $P^+(V)$ are clamped onto the visible units, of finding both unit i and unit j turned on when the Boltzmann machine runs at equilibrium. $< -s_i s_j >^-$ is the averaged probability of finding both unit i and unit j turned on when the Boltzmann machine runs freely at equilibrium.

4.2 RESTRICTED BOLTZMANN MACHINES

Restricted Boltzmann Machines(RBM) basically work as a binary form of factor analysis. In research area, there are vary of different usage methods of Restricted Boltzmann Machines(RBM). Rather than choosing or ranking companies on a constant proportion, workers only indicate to system either they choose a company or not. By this way, Restricted Boltzmann Machines(RBM) estimate to find hidden units. Hidden units layer may define the actuations of these company selections.

Specifically, a Restricted Boltzmann Machine is a hypothetical neural network(we explain it with hypothetical, because these actuations have a probabilistic item.)Items contain of following:

- One layer of visible units(users' job matrix and we pre-set them)
- One layer of hidden units(the latent factors we try to learn)
- A bias unit(whose situation is continually on, and is a way of adjusting for the different inherent popularities of each company).

Moreover, there are a stable connection between each visible unit and all the hidden units. Connection type is undirected, hence there also were a connection between each hidden unit

and the visible units . To provide a simple learning, we limit the connections so that there is no connection between hidden units and visible units.

In detail, Boltzmann Machines (BMs) are a particular form of log-linear Markov Random Field (MRF), i.e., for which the energy function is linear in its free parameters [46]. To make them powerful enough to represent complicated distributions (i.e., go from the limited parametric setting to a non-parametric one).We consider that some of the variables are never observed (they are called hidden variables) [46]. Modelling capability of Boltzmann Machine(BM) can be improved by including more hidden units(hidden variables). Restricted version of Boltzmann Machine(BM) is more narrowed BM to without hidden hidden, visible visible contacts.A node presentation of Restricted Boltzmann Machine(RBM) is detailed below:

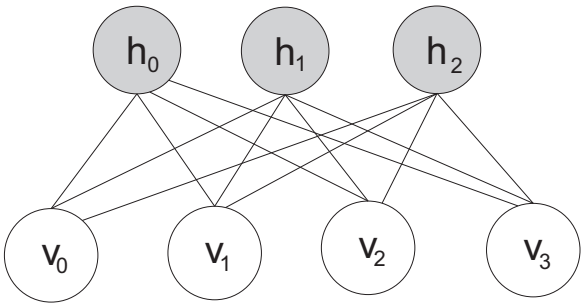


Figure 4.1: A restricted Boltzmann machine

In Restricted Boltzmann Machine(RBM), formulation of energy(energy func.) $F(u, h)$ is explained as:

$$F(u, h) = -a'u - b'h - h'Wu \tag{4.6}$$

In equation 4.7, we show the connection weight of visible and hidden weight with W. a, b are the offset values of hidden and visible layers, mutually.

$F(u, h)$ can be converted to formulation of free energy($E(u)$) in below:

$$E(u) = -a'u - \sum_i \log \sum_{h_i} e^{h_i(b_i + W_i u)} \tag{4.7}$$

As the particular formation of Restricted Boltzmann Machine(RBM), hidden and visible variables are free of each other in some cases.Using variable independence, we are able to propose equation 4.8.(Restricted Boltzmann Machine(RBM) with binary variables)

$$p(h|u) = \prod_i p(h_i|u) \tag{4.8}$$

$$p(u|h) = \prod_j p(u_j|h) \tag{4.9}$$

Generally ,using binary variables(u_j , $h_i \in 0, 1$) is deep study subject.From equations 4.2 and 4.28 we can obtain the probabilistic type of the common neuron actuation formula:

$$p(h_i = 1|u) = \text{sigm}(b_i + W_i u) \tag{4.10}$$

$$p(u_i = 1|h) = \text{sigm}(a_j + W'_j h) \tag{4.11}$$

In Restricted Boltzmann Machine(RBM) with binary variables, formulation of energy(energy func.) $E(u)$ is detailed explained as :

$$E(u) = -a'u - \sum_i \log(1 + e^{h_i(b_i + W_i u)}) \tag{4.12}$$

for updating equations with binary units. Combining equation 4.5 with equation 4.11 ,gradient of log likelihood is obtained for the Restricted Boltzmann Machine(RBM) with binary variables :

$$-\frac{\partial \log p(u)}{\partial W_{ij}} = F_v[p(h_i|u) \cdot u_j] - u_j^{(i)} \cdot \text{sigm}(W_i \cdot u^{(i)} + b_i) \tag{4.13}$$

$$-\frac{\partial \log p(u)}{\partial b_i} = F_u[p(h_i|u)] - \text{sigm}(W_i \cdot u^{(i)}) \tag{4.14}$$

$$-\frac{\partial \log p(v)}{\partial a_j} = F_u[p(h_i|u)] - u_j^{(i)} \tag{4.15}$$

To understand the connection weights in network, we use worker job matrix as training dataset. Each job vector is a binary vector with continuous element size corresponding to a user's job choices. We therefore have to explain the sampling and learning steps in the Restricted Boltzmann Machine(RBM) algorithm.

In this step, we have to sample in Restricted Boltzmann Machine(RBM). To do that, we can get samples using Gibbs method as the conversion controller and we can union $p(x)$ samples by processing the Markov chain.

The Gibbs sampling of the joint of N random variables $S = (S_1, ..., S_N)$ is done through a sequence of N sampling sub-steps of the form $S_1 \sim p(S_i|S_{-i})$, where S_{-i} contains the $N - 1$ other random variables in S, excluding S_i [60].

In Restricted Boltzmann Machine(RBMs), S_i contains the set of hidden, visible variables. Still, a variable is able to get block sample by using Gibbs method, because set of variables are in free case. With this adjustment, we get visible variables sample concurrently obtained the certain values of hidden variables. Likewise, we get hidden variable samples in the same way concurrently. The formulation of Markov chain step for thus taken is :

$$h^{n+1} \rightarrow \text{sigma}(W' u_{(n)} + b) \tag{4.16}$$

$$u^{n+1} \rightarrow \text{sigma}(W h^{n+1} + a) \tag{4.17}$$

In equation 4.16 and 4.17 $h^{(n)}$ shows the $n_t h$ move of the Markov chain and contains all hidden variables at the step n . To explain that, e.g., $h_i^{(n+1)}$ is arbitrary selected to be 1 (versus 0) with probability sigma$(W_i' u_{(n)} + b_i)$, and Likewise, $u_j^{(n+1)}$ is arbitrary selected to be 1 (versus 0) with probability sigma$(W_j h^{(n+1)} + b_j)$.

Graphical illustration can be presented as :

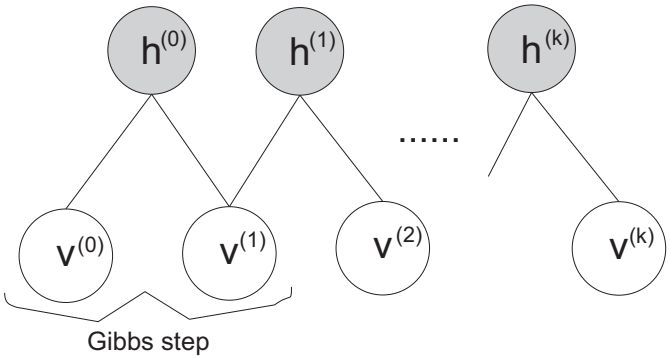


Figure 4.2: Calculating the Gibbs steps in a restricted Boltzmann machine

As $t \rightarrow \infty$, samples $(h^{(t),u^{(t)}})$ are promised to be right for $p(u, h)$.

In learning approach, each update of parameters requires to process one similar chain to union.Thus, it is too costly to operate this process. Intrinsically, various methods have been developed to $p(u, h)$ sampling for learning part of Restricted Boltzmann Machine(RBMs).

To explain learning part,we can extend unions of network (e.g., the failure between the training samples and their constructions) or we reach some maximum number of epochs by utilizing the Contrastive Divergence approach; thereby, helping the network's daydreams better match the reality of our training examples [46].

Contrastive Divergence (Approximate Gradient Descent, CD-k) will be introduced in section 4.3. However, there is no common solution on choosing learning rates. Generally, smaller

learning rates are used for learning both biases. Since RBM is a special case of BM, it is available to use the same Gibbs sampling to learn. Thanks to restricted structure of Restricted Boltzmann Machine(RBMs), Gibbs sampling can be used more efficiently, as given one layer, either visible or hidden, the neurons in the other layer become mutually free.

4.3 TRAINING RESTRICTED BOLTZMANN MACHINE

The learning rules of Contrastive Divergence in RBM , then, become:

$$w_{ij} \leftarrow w_{ij} + \eta_w\Big[\langle v_i h_j\rangle_d - \langle v_i h_j\rangle_m\Big] \tag{4.18}$$

$$b_i \leftarrow b_i + \eta_b\Big[\langle v_i\rangle_d - \langle v_i\rangle_m\Big] \tag{4.19}$$

$$c_j \leftarrow c_j + \eta_c\Big[\langle h_j\rangle_d - \langle h_j\rangle_m\Big] \tag{4.20}$$

where the same shorthand notation $\langle . \rangle_{P(.)}$ was used as before.

Although there is no accurate theoretical background on choosing learning rates, commonly, smaller learning rates are used for learning both biases [39]. Since RBM is a special case of BM, it is available to use the same Gibbs sampling to learn. Thanks to its restricted structure, Gibbs sampling can be used more efficiently, as given one layer, either visible or hidden, the neurons in the other layer become mutually independent (see Figure 4.3). This possibility of the layer-wise sampling enables the full utilization of the modern paralleled computing systems.

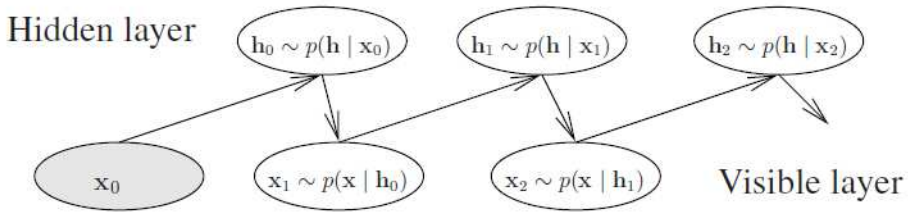


Figure 4.3: Visualization of the idea of how the layer-wise Gibbs sampling is done in RBM.

Notwithstanding, as the size of neurons in RBM increases, a greater size of samples must be gathered by Gibbs sampling in order to accurately explain the probability classification represented by RBM. Moreover, due to the nature of Gibbs samplings, the samples might still miss some modes of the classification.

CD learning approximates the true gradient by replacing the assumption over $P(v, h|\theta)$ with an assumption over a distribution P_n that is gained by running n steps of Gibbs sampling from the

experimental distribution defined by the training samples. Figure 4.4 shows the distributions P_0 and P_n.

For the weights, the CD learning formula, then, becomes :

$$w_{ij} \leftarrow w_{ij} + \eta\Big[\langle x_i h_j \rangle_{P_0} - \langle x_i h_j \rangle_{P_n}\Big] \tag{4.21}$$

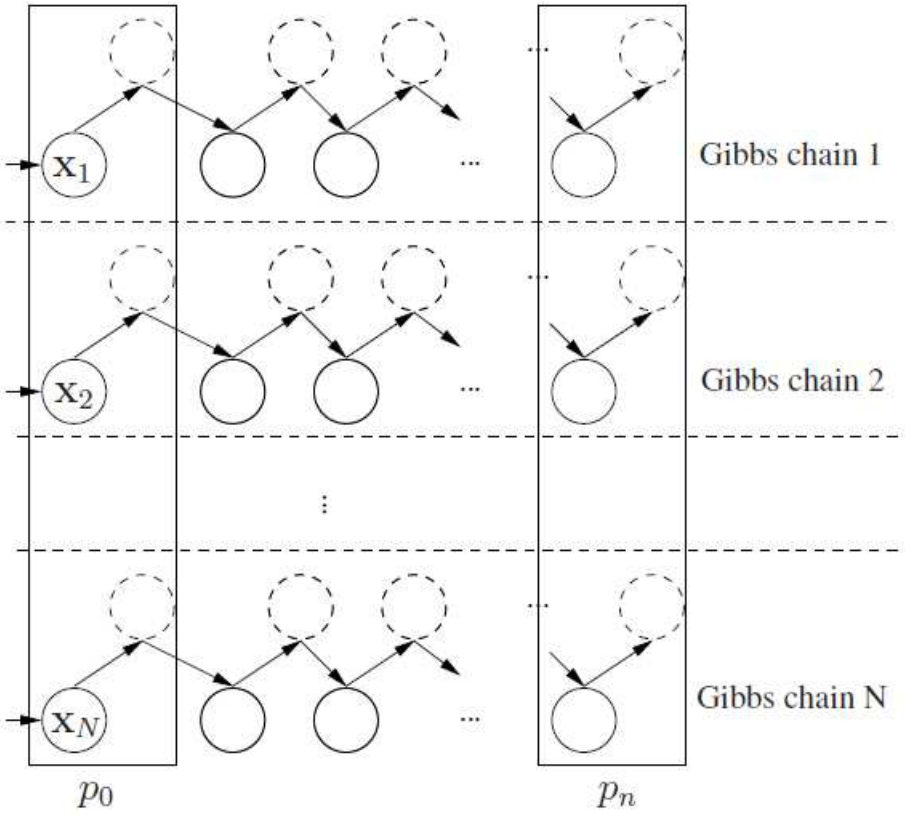


Figure 4.4: Visualization of how CD learning obtains the empirical distribution used in the positive phase and the approximate model distribution used in the negative phase.

It should be mentioned that the case $n = 0$ produces the experimental distribution $P(h|\{v^{(t)}\}, \theta)$ used in the positive part, whereas the case $n = \infty$ produces the true distribution of the negative part $P(x|\theta)$ [42] As it can be assumed from the fact that the direction of the gradient is not identical to the exact gradient, CD learning is known to be biased [42]. However, CD learning has been shown to work well in practice. A nice property of CD is that in case the data distribution is multi modal, running the chains starting from each data sample guarantees, that the samples approximating the negative part have exemplar from different modes.

4.4 RECOMMENDATION APPROACH

Imagine for an instance that the distribution of patterns of the previous example corresponds to user ratings instead of geometric shapes. A pixel would correspond to a company and its intensity to the rating the user gave to the company. Accordingly, learning the distribution of patterns would amount to learning how to perfectly regenerate the entire dataset of ratings. Of course, this may not be realistic on a dataset as large as our dataset, but it could still be a reasonable thing to do since it would identify dependencies between choices and companies. Then, just as we used RBMs to reconstruct incomplete and scrambled geometric shapes, the rating pattern of a user could hopefully be completed with the companies he would most likely appreciate. In essence, this is the strategy we use throughout the remainder of this work.

The model presented in this section was first proposed in [47] by Ruslan Salakhutdinov et al. Assume that we have M company and N users and that the relations are binary with time scale. An easy solution to this problem is to use composite visible units, called *softmax* units, and which roughly comprise a combination of K binary visible units. Each time-based choice is transformed into a binary code such that the $k-th$ binary unit of the softmax is turned on if and only if the user worked that company as k. The second problem is how to deal with the large number of missing choices. The solution proposed in [47] is to consider that the visible units corresponding to the company at which the user did not work simply do not exist. In practice, this simply amounts to consideration that these visible units are always turned off and hence that their state is always zero. Alternatively, this can also be seen as using a unique RBM per user, all sharing the same weights and biases, all with the same number of hidden units but each only including the softmax units for the companies worked by their user.

In common, RBMs and neural networks operate by changing the conditions of similar neurons given conditions of other neurons; therefore, in this situation, we can examine how the conditions of exclusive variables change. After determining the link weights, to change the condition of variable i is described the following steps in our RBM :

Firstly, we calculate the actuation energy $a_i = \sum_k w_{ik} x_k$ of variable i, where the sum runs over all variables k that variable i is linked to, w_{ik} is the weight of the link between i and k, and x_k is the 0 or 1 condition of variable k. Namely, all of variable i's neighbours send it a info, and we calculate the sum of all these infos. Subsequently, let $p_i = \sigma(a_i)$, where $\sigma(x) = \frac{1}{(1+\exp(-x))}$ is the logistic formula. Notice that p_i is near to 1 for large positive actuation energies, and p_i is near to 0 for negative actuation energies. We next change variable i on with probability p_i, and change it off with probability $1 - p_i$. (Positively linked variables attempt to share the similar condition, while negatively linked variables chose to be in separate conditions. So, we can entitle negative ones as encounter-er)

Hence, we can find the following solution to learning problem of the link weights. We have a number of training samples, in which each training sample is a binary vector with continuous elements corresponding to a worker's choice options. For each epoch, we take a training sample (a set of company options) and set the conditions of the visible variables to these options.

Next, the conditions of the hidden variables are changed using the logistic actuation rule described for the kth hidden variable. We next calculate its actuation energy $a_k = \sum_i w_{ik} x_i$, and set x_k to 1 with probability $\sigma(a_k)$ and to 0 with probability $1 - \sigma(a_k)$. next, for each edge e_{ik}, we calculate Positive $(e_{ik}) = x_i \cdot x_k$ (i.e., for each couple of variables, measure either they are both on). We next reconstruct the visible variables in a similar manner: for each visible variable, its actuation energy a_i is calculated, and condition is changed. (Notice that this reconstruction may not match the original options.) Next we change the hidden variables again, and calculate Negative $(e_{ik}) = x_i \cdot x_k$ for each edge. The weight of each edge e_{ik} is prepared by setting $w_{ik} = w_{ik} + L \cdot (Positive(e_{ik}) - Negative(e_{ik}))$, where L is a learning rate. This process is repeated to finish over all training samples [48]. The algorithm ends when it reaches maximum number of epochs or completes network unions.

Algorithm 4 RBM - Making recommendations

Inputs: a user i, a company j
Outputs: an estimation of $R(i, j)$
Clamp the ratings of i over the k(softmax) units of the RBM.
Compute $a_i = \sum_j w_{ij} x_j$ of unit i.
Compute $a_j = \sum_i w_{ij} x_i$ for all hidden units j.
for each edge **do**
  Calculate the *Positive* and *Negative* measure for units.
  Change the weight of edge e_{ij} by operating $w_{ij} = w_{ij} + L \cdot (Positive(e_{ij}) - Negative(e_{ij}))$
end for
Take the expectation as the prediction, i.e., $R(i, j) = \sum_{k=1}^{K} (a_i^k = 1 | a_j) k$

Finally, note that this extension of the model happens to be even more interesting in the context of our recommender systems. Indeed, in that case, r could similarly be populated with the (much larger) list of companies a user worked at (instead of only those at which he is working). Again, we would not know whether that is so and we cannot infer a working prediction from only this source information, but it may still constitute a very good head start.

4.5 EVALUATION AND COMPARISON

Evaluations were again conducted using off-line analysis, a variety of user experimental methods with train and test datasets, and a total combination of these. Much of the work in algorithm evaluation has focused on off-line analysis of predictive accuracy. In such evaluations, the algorithm is used to predict certain withheld values from a dataset, and the results are analyzed using one or more of the metrics [32].

For the accurate comparison of algorithms, we take a so-called backtesting approach again to evaluation that is common in recommender systems literature [5]. Whether backtesing had been used in the time of former period, this evolution method tries to classify the efficiency

of a approach.

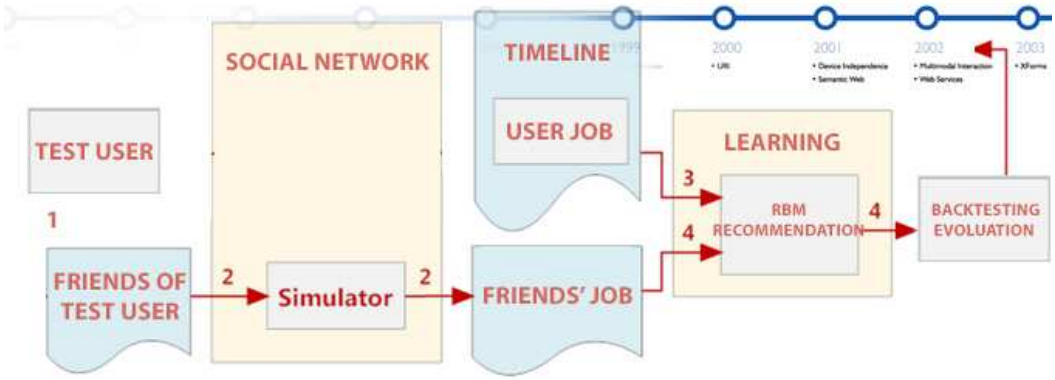


Figure 4.5: The flow diagram of recommender methodology in RBM and backtesting steps.Diagram contains simulation of recommendation test and evaluation steps.

In the test method, a set of test workers are examined, then a single test interval is examined, without the jobs being hidden after the experienced test interval for each test-worker. This approach simulate test condition of recommendation system initiated for two different methodologies, and after makes suggestions without considering any new data that comes later from the first interval. Considering that interval is an another option. At start, set of test workers are taken for sampling, after the number n_u of jobs to hide for each worker u are taken for sampling, after that n_u jobs to hide are taken too. Thus, our approaches presumes that this temporary situation of worker selections is trivial. All next different selections divide the dataset to only one test and training group. Choosing a way that is most proper for our concern and research interest is necessary for the given constricts. Taking into consideration our dataset, and constraints of off-line analysis, we chose the second approach to compare our results with that of the item-to-item recommendation method.

Algorithm 5 RBM - Test algorithm

1. For all user-company pairs in the test set T , compute $R(i, j)$ using algorithm 4.
2. Compute the RMSE of the predictions:

$$RMSE = \sqrt{\frac{1}{|T|}\sum_{(u,i\in T)}(\hat{r}_{ui} - r_{ui})^2}$$

Correctness of prediction is mostly separated from the worker profile, and so is calculated in an off-line test. Checking correctness of prediction in a worker aspect calculates the correctness given a recommendation. This approach is closer to the true correctness for the on-line application and is a different approach from the guess of worker behaviour without giving any recommendation.

We employ two famous formulas to calculate the correctness and the prediction quality of our

propositioned method (MAE - Mean Absolute Error, RMSE - Root Mean Square Error)[50] and to contrast it with another recommendation algorithm. In addition to metrics, ROC Curves are evaluated with AUC values. These techniques are described and detailed in Chapter 3.

4.5.1 EXPERIMENTAL RESULTS

Results are separated in two parts: performance and quality. For quality of predictions, initially parameter sensitivity is determined before processing the main test. These parameters contain the count of K-Means root, the impacts of various distance measures, similarity measures and rank of the friendship ratio x to determine the sensitivity

In this section, we present our experimental results obtained after applying Restricted Boltzmann Machines to generate recommendations. Results are separated to two parts: performance and quality. For quality of predictions, initially parameter sensitivity is determined before processing the main test. These parameters contain the count of K-Means root, the impacts of various similarity measures and rank of the friendship ratio x. To determine the sensitivity of different parameters, we focus on the off-line evaluation result and further separated it into different sparsity parts, which we used to test the parameters. First of all, the motivating results are explained, after which we describe the effect of the item-to-item algorithm and the RBM difference.

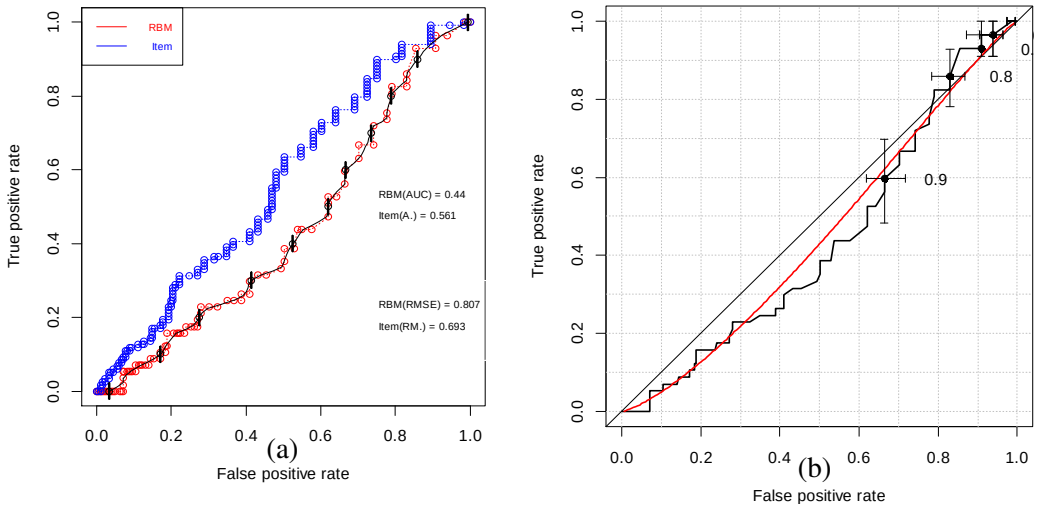


Figure 4.6: (a) Comparison of the ROC Curves (with inclined form), AUC, and RMSE predicted by our models against RBM and Item-to-Item Algorithm using FMF social measure. Over 300 users tested with minimum 20 friendship sizes.(b) Average and cut-off points of two ROC Curves produced by evolution algorithm.

We first tested the social network measures' clustered results with only one distance function similarity against the others. We then compared the RBM results to each other using friend's distance clustered subset. A presumption of binary relevance is generated from (ROC)curves : Recommended jobs are classified successful(relevant) or unsuccessful(non-relevant) and that can be assigned 1 and 0. As a result of this presumption the ordering among related jobs has no concern on the (ROC)curve metric.

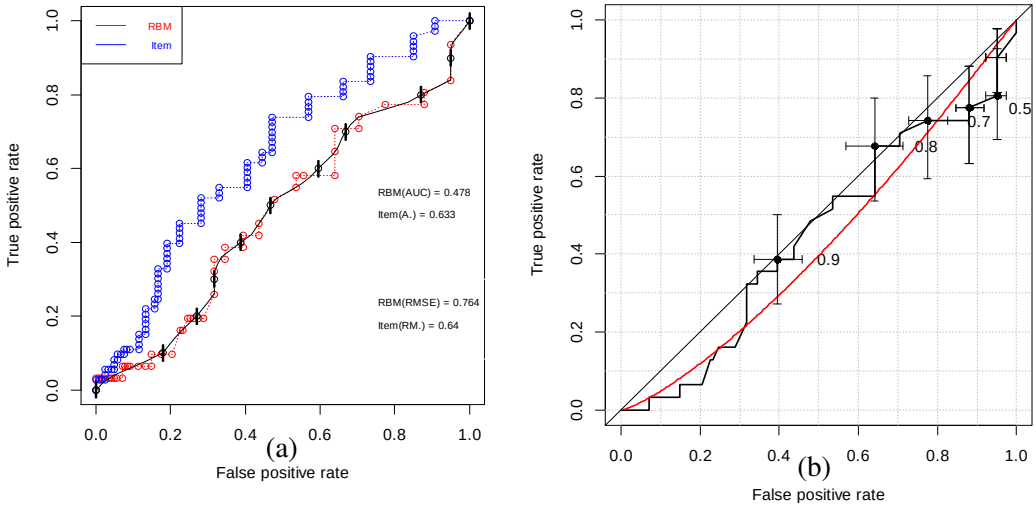


Figure 4.7: (a) Comparison of the ROC Curves (with inclined form), AUC, and RMSE predicted by our models against RBM and Item-to-Item Algorithm using WF social measure. Over 300 users tested with minimum 20 friendship sizes. (b) Average and cut-off points of two ROC Curves produced by evolution algorithm.

We mark the hit/miss rates on x - y coordinates, mutually. Different thresholds build a serial of points and the shape of serial is like a curve. In the literature, the area under curve(AUC) is a performance measure, with perfect performance indicated by an area of one and random guessing indicated by an area of 0.5 [38]. Figure 4.6 compares the size of this area to a perfect curve in order to determine the better quality of clustering and recommendation of our item-to-item method.

The experiments were all performed on our dataset with the same user. The ratings from the probe set were all extracted from the original item training set, resulting in a test dataset of an 82,000-worker company. Models were all trained for 300 or 1000 passes (or epochs) on that dataset. The training error of the model was computed (c.f., Algorithm 5) after each epoch on a random subset of the training set.

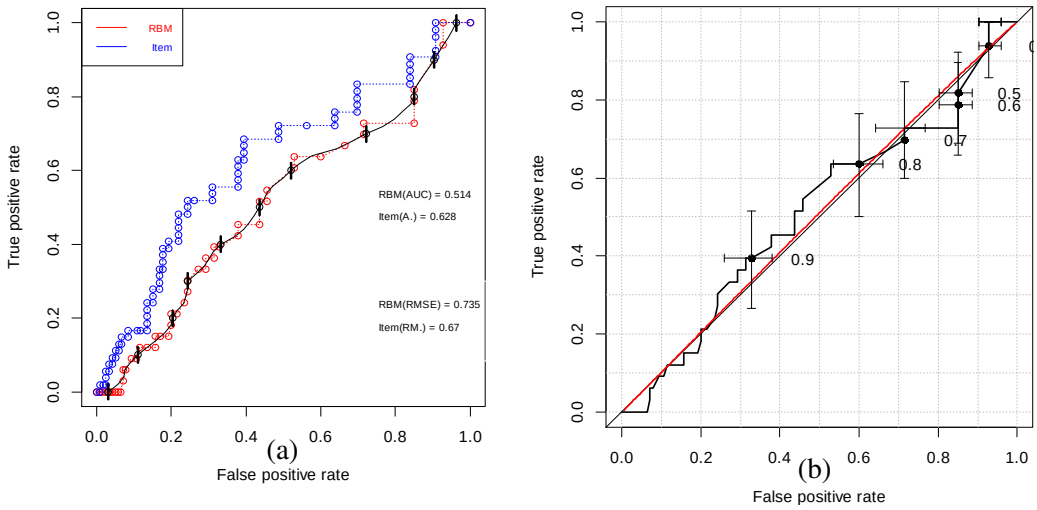


Figure 4.8: (a) Comparison of the ROC Curves (with inclined form), AUC, and RMSE predicted by our models against RBM and Item-to-Item Algorithm using MF social measure. Over 175 users tested with random friendship sizes.(b) Average and cut-off points of two ROC Curves produced by evolution algorithm.

The generalization error (i.e., the error on an independent dataset) was evaluated on the probe set. Learning was not terminated if the model started to over-fit the training set (in contrast to the stop criterion of algorithm 4).

We compared different models based on their performance on the validation set. The error from our dataset on the test set was commonly larger than the error we got on the validation set. When the validation set was added to the training set, RMSE on the test set was commonly reduced by about 0.029. If we subtract RMSE in Figure 4.7 from RMSE in Figure 4.8, we eventually acquire reduced value. Figure 4.6 (left panel) presents the performance of the RBM and the RBM with more friendship size. It is clear that the more friendship-tested model considerably exceed its linear counterpart. Figure 4.6 (right panel) also reveals that conditioning on weighted edge information significantly improves model performance.

Figures 4.6, 4.7, and 4.8 also show the results for three different social measure clustering against the RBM, AUC, and RMSE results: Test on Friend Distance Measure (FD), Test on Mutual Friend Measure (MF), and Test on Friend's Mutual Friends Measure (FMF), Weighted Friend (WF). All accuracy measures were collected over 10 randomly cross-validated runs. It can be seen that the **FD** output with Item-to-Item Method achieves a significant boost over the RBM.

As noted before, increasing the number of nodes directly increases the representational power of the model. As Figure 4.8 indeed illustrates, the greater the number of nodes, the further

the RMSE is reduced over the training set. On the probe set however, as Figure 4.7 shows, increasing the number of hidden nodes beyond 30 to 1000 does not make the model any better in general. Quite logically, the greater the number of user nodes, the more the model over-fits the data and the worse it becomes in general.

Table4.1: Accuracy values of RBM and Item-to-Item algorithms in graphs 4.6, 4.7, and 4.8

	Hit Rate	Miss Rate	Non Rec.	AUC	RMSE
RBM-1	%20.34	%70.6	%9.89	*0.440*	*0.807*
Item-1	%32.37	%66.02	%2.4	*0.561*	*0.693*
RBM-2	%16.61	%80.43	%4.98	*0.478*	*0.764*
Item-2	%28.74	%70.53	%4.33	*0.633*	*0.640*
RBM-3	%26.37	%62.34	%2.88	*0.514*	*0.735*
Item-3	%34.14	%52.53	%2.78	*0.628*	*0.667*

Finding a satisfying combination of parameters is not an easy task. Firstly, all of them have a specific effect on the learning curves. Secondly, they all add up together on the final result, sometimes canceling or reinforcing each other, which complicates matters even more. Thirdly, given the time required to evaluate a set of parameters, it is not practical to test every possible combination. In that context, the three experiments presented below only aim at studying the individual effect of each parameter, when all other things are kept equal. Simulations were all carried out using the conditional model.

Table 4.1 also shows the results for all similarity measures used for the item-to-item filter algorithm and RBM hit and miss rate results. All accuracy measures were collected over the same schema and after clustering all methods.

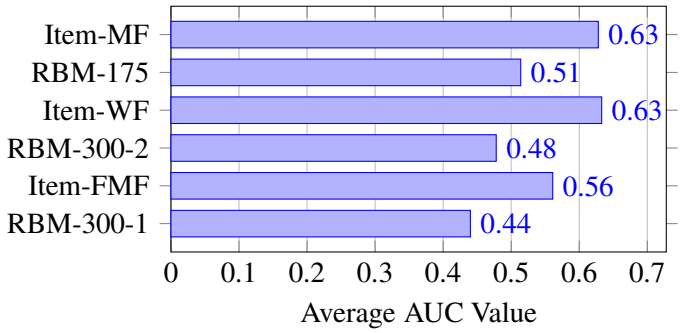


Figure 4.9: Comparison of AUC values prediction on all RBM tests

4.6 RUN-TIME PERFORMANCE AND APPLICABILITY

We introduced a class of two-layer undirected graphical models (RBMs), suitable for modeling tabular or count data, and presented efficient learning and inference procedures for this class of models. We also demonstrated that RBMs can be successfully applied to a large dataset containing over 1.4 million worker/company choices.

We ran our experiment on a 12-processor Nehalem server machine. All processors were Xeon 2.8 GHz and every four processors shared 6 GB of RAM. For each parameter and function change, we ran 4-8 rounds of our algorithm and stopped when one round of U, M update improved the RMSE score on the probe dataset. The test RMSE was obtained by writing to the thesis. The true values of the test ratings are known to us; for model building and parameter tuning, we excluded the probe dataset from the training dataset and used it for testing. The probe dataset is a subset of the training dataset, provided by our web crawler, and consists of 1,408,395 of the latest choices in year 2009, while users are sampled uniform random and at most all choices are drawn for each test user.

Table4.2: Overall performance values of RBM

	Avg. Execution Time
RBM-300	7.22 s
Item-300	4.65 s
RBM-175	1.62 s
Item-175	0.94 s

The experimental results obtained by applying various collaborative filtering techniques to generate predictions are presented here. Results are separated to two parts: performance and quality. For quality of predictions, the type of clustering measure is determined before processing the main test. In Restricted Boltzmann Machines, the type of distance density and friendship size is tested at each step. This different type of determination shows the quality of our new approach.

After showing that the item-based algorithm provides better quality prediction than Restricted Boltzmann Machines, we focus on performance issues. As discussed before, item-based similarity is more static and allows us to pre-compute the item neighbourhood. However, RBM is more dynamic to sparsity of dataset and size of hidden units. To make the system even more scalable, we examine them and investigate the impact of the friendship size on the CPU response time.

Table 4.2 shows the time measures of the algorithms for three different RBMs on the CPU and their comparison. Results prove that proposed algorithm does better concerning variety, quality and has not a loss in success, despite that it is more scalable. Also note that, the

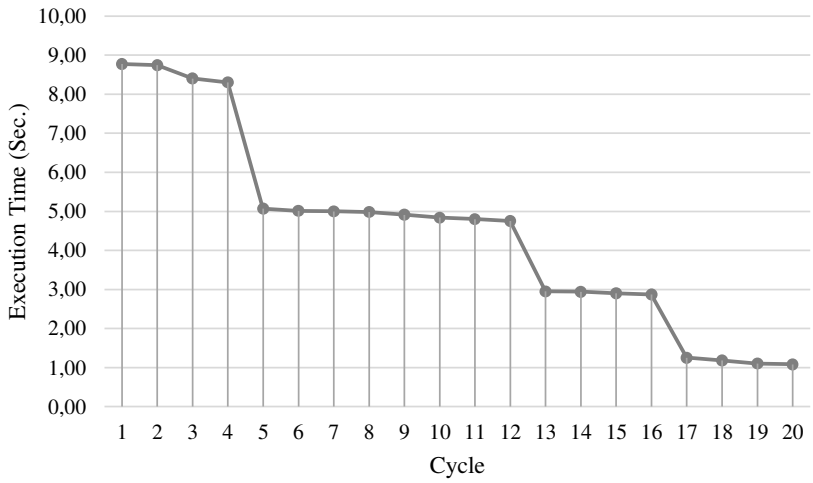


Figure 4.10: Distribution of execution time with various counts of user and hidden units. Algorithm tested with a total of 20 execution cycles.

diversity in time does not change when the number of friends increases. This point is key to scalability.

As discussed earlier, RBM is more dynamic to sparsity of dataset and size of hidden units. This comparison of the model has certain performance disclosures. Figure 4.10 illustrates that all off-line test combinations consume similar CPU time but lower the quality of the prediction. Our test environment and recommendation methodology is a reasonable scale for RBM tests. To make the system even more scalable, we looked in detail at the item-to-item method specificity of the model and investigated the effect of model size on the throughput and response time.

4.7 DEMONSTRATION OF IMPLEMENTATION AND EVALUATION

Our approaches to collaborative filtering are demonstrated in section 3.8. In the figure 4.10, we show how a class of two-layer undirected graphical models, called Restricted Boltzmann Machines (RBMs), can also be used to model user's choice of job. In the screenshot(a), the test algorithm chose a previous job in her/his working timeline and the prediction is computed from the past employment of them. To find people who are similar to a user, the system divides the users' friends (target set) into subsets, and treats the task as a RBM learning problem. We present efficient learning and inference procedures for this class of models and demonstrate that RBMs can be effectively applied to our dataset for comparison in screenshot(b).

The algorithm finds similar companies for which the target people have worked, aggregates those companies, and then recommends the most popular or correlated companies in screen-

shot(c).

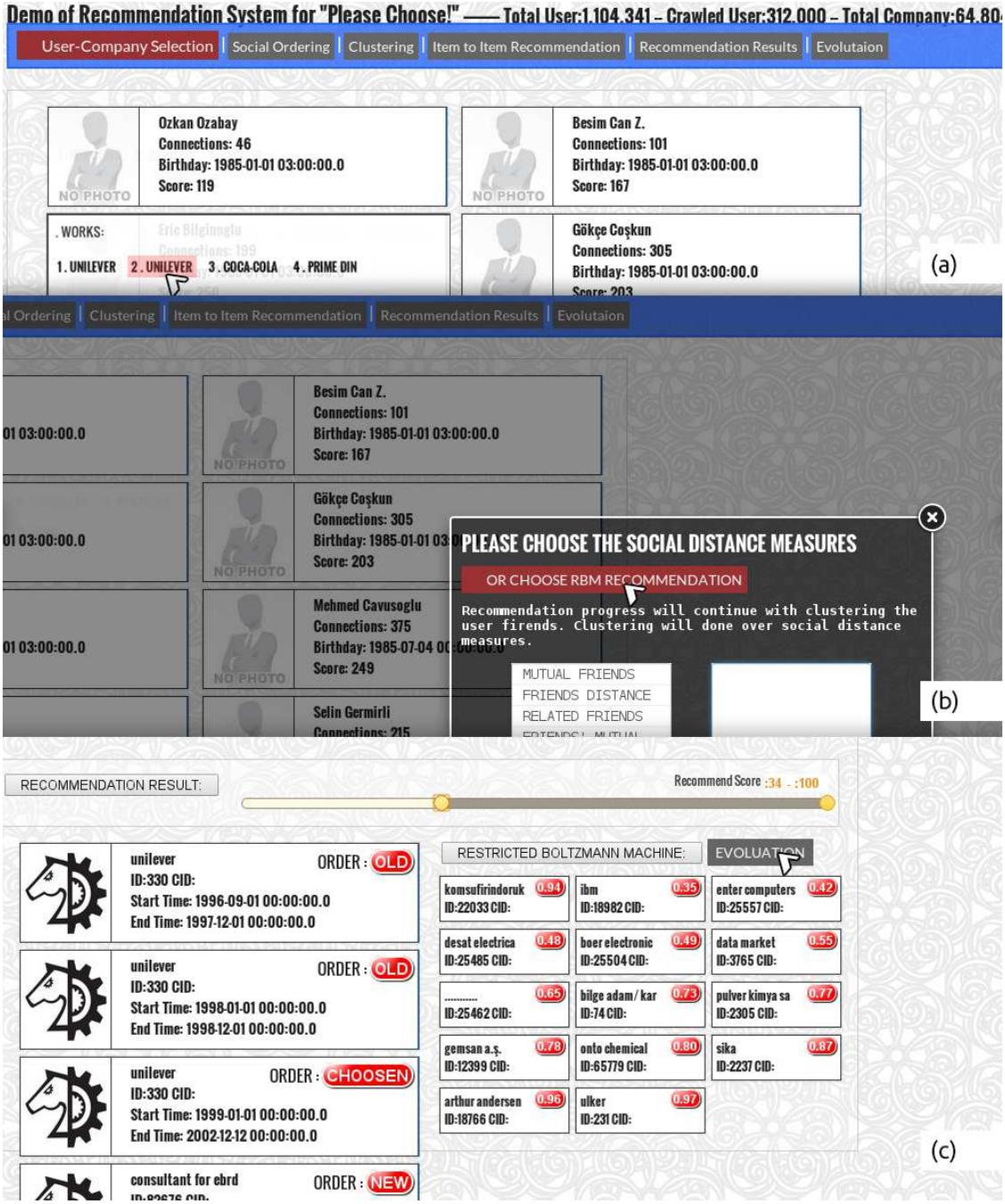


Figure 4.11: Screenshot(a) shows job choice for the first step of backtesting; (b) presents main recommender selection for RBM; (c) displays the ordered job recommendations and its evolution metrics.

We are able to compare recall to precision, or false positive rate to true positive rate and backtesting results. The curves of the previous version are identified commonly as the curves

of precision-recall, while those of the next version are identified as ROC (Receiver Operating Characteristic curves). Information related to present and past employment data allows us to draw ROC curves and calculate RMSE values after recommendation. While top-N limits the predictions to some predefined number, ordered search results allow the worker to continue to look at jobs highly likely to be of interest to them. This feature allows workers to have query returns sorted by the predicted likelihood that the worker will chose the job. Once again, this helps convert workers into employees.

4.8 DISCUSSION

We found that combining different recommendation runs yields more competently performance compared to the singular runs, which is consistent with the theory behind social synthesis and with the related work. Item-to-item recommendation method consistently outrun non-collaborative complements. This is not surprising as it is unlikely that every run contributes equally to the final result, and this was also obvious from the optimal advise distribution among the runs.

We explored two particular aspects to recommendation, representation and the choice of algorithm, and really found that runs that combine multiple, different recommendation aspects perform more competently than runs that consider alternative in only one recommendation aspect.

In the experimental evaluation of our approach, we have focused on using wide collected data sets and comparing our approach against state-of-the-art approaches, something which is lacking from much of the relevant approach. We are aware, yet, that our comparisons are by no means complete as we picked only two promising approaches to compare our approach with. Model-based collaborative filtering algorithms, for example, were lacking from our comparison. In 'ordinary' recommendation experiments in different domains, model-based algorithms have been shown to hold a slight edge over memory-based algorithms, but without proper comparison on professional social network data sets we cannot deduce any conclusions about this. One natural extension of the approach would accordingly be to extend the comparison we made to all of the relevant approaches discussed in Chapter 1.3. Such a comparison would have to include at least the approaches by [1], [5], [14] and [25].

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 CONCLUSION

In this thesis, we presented algorithms for a scalable real-time recommendation engine and examined the results of evaluations conducted on a professional social network. The recommendation within the social networks requires different approach then these in the regular recommender systems. The goal of the suggestions within social networks is to stimulate the evolution of the networks and to create the relationships that will be permanent. Social network measures enables to influence the way in which the social network will evolve. This makes the framework flexible. Moreover, we performed a principled investigation of the usefulness of different algorithms and information sources for recommending relevant jobs to workers of professional social network. Below we list the following four conclusions we have made :

- We examined different ways of using the information present in a social network for recommendation, by extending a standard class of Collaborative Filtering algorithms with information from the social network. These extensions were then compared to other RBM approach, and shown to be competitive.
- We determined the best way of using item-to-item methodology for recommendation, and proposed several new and hybrid algorithm. This algorithm were shown to be competitive with the more popular usage-based approaches that use the social network.
  - We clearly demonstrate that item-to-item scheme provides better quality of predictions than RBM scheme in this domain. Besides, The RBM is a good method to automatic model a data distribution and RBMs show really good results for recommendations too.
- Compared to related work, we took a critical look at different methods for combining recommendations steps on the same data set. We showed that combining collaborative filtering and social clustering , that all cover different aspects of the recommendation task, yields the best performance, confirming earlier work in the field of IR.

  - With these combinations, we are able to claim that item neighbourhood is fairly static and it can be pre-computed it produces results in very high on-line performance. But, RBM computational more expensive than proposed algorithm as a result of learning steps.

- The recommendation of one human being to another is much more complicated that generation of the suggestion of jobs. The reason for that is the bidirectional character of workers' relationships. The aim is not only to find the person y that will match worker x propensities, but also user x must suit member's y propensities. Social clustering gives the opportunity to analyse the social aspects of acquaintances such as the maximum number of relationships that can be actively maintain by one human being.

  - We obviously understand that it is possible to retain only a small subset of jobs with social clustering and produce reasonably good prediction quality. Also, analysis of social graphs in collaborative filtering applications improves quality of recommendations.

In Chapter 2, we first reviewed the main categories of recommendation algorithms. The most popular and effective approaches, namely collaborative algorithms, were then studied in more detail in the second half of the chapter. Chapter 3 was devoted to our item-to-item collaborative filtering algorithm. The goal of the research was to substantially improve the recommender system that our dataset uses to make recommendations to its worker. All in all, the challenge was a great success and highly beneficial to the science of recommender systems. Many new ideas and algorithms have been proposed by hobbyists and researchers on this subject.

In Chapter 4, a class of machine learning models called Boltzmann machines was reviewed in depth, first from a general point of view, and then in the context of recommender systems. A full implementation of the model was written and then experimentally tested on our dataset. The results obtained, in terms of accuracy over an independent test set were very satisfying and were close to, and at times surpassed, some of the results published in the literature. One of the strongest issues in the experiments described in Chapter 4 was that in order to obtain satisfying results computing times sometimes had to be long. In that context, three different approaches were proposed in Chapter 5 to make Boltzmann Machines more scalable.

Ideally, we would like to try all ideas off-line, evaluate them, and only push to production the best ones. A hybrid methodology proved critical in the evaluation step. Over time, using implicit and explicit feedback combined with social network measures, the system accumulated a huge gold test set for user profiles. At this point, using standard ranking metrics and parameter choosing can rank the gold set with each model on commonly used Machine Learning techniques to evaluate which one performs best. That approach is another research issue and *LinkedIn* developers claim that their filtering methodology consists of similar features.

5.2 FUTURE WORK

We propose a set of actions to continue the work done in the thesis. Addtionally to the implementation, we propose some more activities in the recommender research.

We can claim that our recommendation system will add a powerful new dimension to a web applications by driving users to other products or web offerings, based on their social characteristics or behaviour. This thesis provided a brief introduction of techniques used by recommendation part of an application, and what developers would need to do to make them more scalable. We also show how recommendation algorithm leverages these techniques and helps you integrate them into a an real-time application. By integrating these concepts into movie recommendation system, we learned that we can extend our applications and add effective personalization to them.

For future implementation, we will use our recommendation system in a web application which enables customers to set up their own music store, based on albums and artists they like . Customers will be able to indicate which albums they own, and which artists are their favourites. Purchases from the web application will be entered automatically into the list. Although list ratings are initially treated as an indication of positive likes, When customers request recommendations the system will predict list of albums the customer might like based on what is already owned. A social recommend option can be added by friends of customers providing a positive or negative comment for any of the albums in their prediction list. The albums recommended change based on the social network and social feedback.

Moreover, as mentioned above, we will integrate our methodology to predict recommends movies to customers based on their previously indicated interests with proposed recommendation algorithm. Customers will enter a rating on a a scale for movies they have viewed. These ratings will be used for item choice and as they continue, the information page for non-rated movies contains a socialized prediction. In a variation of this, customers can use social recommendation to search for top picks based on social profile and choose to have these sorted by their socialized prediction or by the all friends average.

We have presented the implementation of an item-to-item recommender system. In this system, one first pre-computes popularity differentials in a matrix from which associated and socialized recommendations can be computed on-line. Also, we design our algorithm easy to implement, maintain and all aggregated methods is easily interpreted by the domain expert. Based on our experience with the out LinkedIn dataset and implementation approach, we can say that this approach gives sensible recommendations and is reasonably easy to integrate to movie and music domains. Hence, by encouraging their users to create a social network, these domains can benefit from providing recommendations based not only on past purchase behaviour, but also based on their customers' social behaviour.

REFERENCES

[1] B. Sarwar, G. Karypis, J. Konstan, and J Riedl. Item-based Collaborative Filtering Recommendation Algorithms . *WWW10, 10TH International World Wide Web Conference*.2011

[2] J. MacQueen. Some methods for classification and analysis of multivariate observations.*Pages 281-297 of: Fifth Berkeley Symposium on Mathematics„Statistics and Probability*.1967

[3] Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. GroupLens: An open architecture for collaborative filtering of netnews. *Pages 175-186 of: Proceedings of the ACM Conference on Computer Supported Cooperative Work*.1994

[4] U. Shardanand and P. Maes. Social information filtering: Algorithms for automating word of mouth's. *Pages 210-217 of:Proceedings of the Conference on Human Factors in Computing Systems Work*. 1995

[5] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. *Pages 210-217 of:Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*. 1998

[6] G. Linden, B. Smith, and J. York. Amazon.com Recommendations - Item-to-Item Collaborative Filtering. *Amazon.com*. 2003

[7] R. Sinha and K. Swearingen. Comparing Recommendations Made by Online Systems and Friends. *Pages 466 of:Recommender Systems Handbook*. 2006

[8] Netflix CineMatch. *netflix.com*. Stated in January 2013.

[9] Netflix Prize. *netflix.com*. Stated in January 2013.

[10] A. S. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: scalable online collaborative filtering. *Pages 271-280 of: WWW '07: Proceedings of the 16th international conference on World Wide Web* .2007

[11] Del.icio.us. *delicious.com*. Stated in January 2013.

[12] J. He and W.W. Chu, A Social Network-Based Recommender System (SNRS). *Pages 47-74 of: In Proceedings of Data Mining for Social Network Data.* 2010

[13] Last.fm Audioscrobbler. *last.fm*. Stated in January 2013.

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

[15] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Pages 734-749 of: IEEE Trans. on Knowl. and Data Eng., 17(6)*. 2005

[16] J. B. Schafer, J. Konstan, and J. Riedi. Recommender systems in e-commerce. *Pages 158-166 of: In EC '99: Proceedings of the 1st ACM conference on Electronic commerce,ACM* . 1999

[17] P. Resnick and H. R. Varian. Recommender systems. *Pages 56-58 of: Commun. ACM, 40(3)*. 1997

[18] M. Balabanovic and Y. Shoham. Fab: content-based, collaborative recommendation. *Pages 66-72 of:Commun. ACM, 40(3)*. 1997

[19] H. Kaya. Using Social Graphs In One-Class Collaborative Filtering Problem. *M.sc. Thesis Metu, Ankara*. 2009

[20] I.S. Dhillon, D.S. Modha. Concept decompositions for large sparse text data clustering. *Technical Report RJ 10147, IBM Almedan Research Center*. 1999

[21] G. Young and A. Householder. Discussion of a set of points in terms of their mutual distances. *Pages 19-22 of: Psychometrika, 3*. 1938

[22] Y. Oike, M. Ikeda, and K. Asadat. A Word-Parallel Digital Associative Engine with Wide Search Range Based on Manhattan Distance. *IEEE 2004 Custom Integrated Circuits Conference*. 2004.

[23] J.B. Kruskal. Nonmetric multidimensional scaling: a numerical method. *Pages 115-129 of: Psychometrika 29*. 1964

[24] B. M. Sarwar, J. A. Konstan, A. Borchers, J. Herlocker, B. Miller, and J. Riedl. Using Filtering Agents to Improve Prediction Quality in the GroupLens Research Collaborative Filtering System. *In Proceedings of CSCW '98*. 1998

[25] N. Good, B. Schafer, J. Konstan, A. Borchers, B. M. Sarwar, J. Herlocker and J. Riedl. Combining Collaborative Filtering With Personal Agents for Better Recommendations. *Pages 439-446 of: Proceedings of the AAAI-'99 conference*. 1999

[26] B. M. Sarwar, G. Karypis, J. A. Konstan and J. Riedl. Application of Dimensionality Reduction in Recommender System-A Case Study. *In ACM WebKDD 2000 Workshop*.2000

[27] G. Karypis. Evaluation of Item-Based Top-N Recommendation Algorithms. *Technical Report CS-TR-00-46*. 2000

[28] A. Rajaraman and J. Ullman. Mining of Massive Datasets. *Cambridge Press 2nd*. 2012

[29] S. Macskassy and F. Provost. A Simple Relational Classifier. *Proceedings of the KDD-2003 Workshop on Multirelational Data Mining*. 2003

[30] Linked.in profile pages. *linkedin.com*. Stated in January 2013.

[31] G. Louppe. Collaborative filtering Scalable approaches using restricted Boltzmann machines. *M.sc. Thesis University of Liege*. 2009

[32] G. Shani, A. Gunawardana. Evaluating recommendation systems. *Pages 257-297 of: Recommender Systems Handbook*. 2011

[33] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. *Proceedings of the Conference on Research and Development in Information Retrieval*. 1999

[34] C. J. Van Rijsbergen. Information Retrieval. *Dept. of Computer Science,University of Glasgow*. 1979

[35] D.C. Bamber. The Area Above the Ordinal Dominance Graph and the Area Below the Receiver Operating Characteristic Graph. *Pages 387-415 of: Journal Math. Psychol.12*. 1999

[36] B. Mason,B. Simon, J. Graham and E. Nicholas. Areas beneath the relative operating characteristics (ROC) and relative operating levels (ROL) curves: Statistical significance and interpretation. *Pages 2145-2166 of Quarterly Journal of the Royal Meteorological Society (128)*. 2002

[37] B. Hanczar, H. Jianpingi ,C. Sima ,J. Weinstein,M. Bittner, M. Dougherty and R. Edward. Small-sample precision of ROC-related estimates. *Pages 822-830 of Bioinformatics 26 (6)*. 2010

[38] B. Hand,J. David. Measuring classifier performance: A coherent alternative to the area under the ROC curve. *Pages 103-123 of: Machine Learning, 77*. 2009

[39] G. E. Hinton, S. Osindero and Y. Teh. A fast learning algorithm for deep belief nets. *Pages 1527-1554 of: Neural computation 18, 7* . 2006

[40] Y. Bengio, Y. Lecun. Scaling learning algorithms towards AI. *Large-Scale Kernel Machines*. 2007

[41] H. Larochelle, Y. Bengio,J. Louradour and P. Lamblin. Exploring strategies for training deep neural networks. *Pages 1-40 of The Journal of Machine Learning Research 10*. 2009

[42] D. H. Ackley, G. E. Hinton and T. J. Sejnowski. A learning algorithm for Boltzmann machines. *Pages 522-533*. 2006

[43] S. Roweis. Boltzmann Machines. *Lecture Notes*.1995.

[44] G. E. Hinton. What kind of a graphical model is the brain? *Pages 1765-1775 of: In IJCAI'05: Proceedings of the 19th international joint conference on Artificial intelligence*. 2005

[45] G. E. Hinton. Boltzmann machine. *Page 1668 of Scholarpedia 2, 5*. 2007

[46] Y. Bengio. Learning deep architectures for AI. F*oundations and Trends in Machine Learning*. 2009

[47] R. Salakhutdinov, A. Mnih, G. E. Hinton. Restricted Boltzmann machines for collaborative filtering. *Page 798 of : In Proceedings of the 24th international conference on Machine learning, ACM*. 2007

[48] G. E. Hinton. A Practical Guide to Training Restricted Boltzmann Machines. *UTML TR 2010-003*. 2011

[49] G. E. Hinton,T. J. Sejnowski. Learning and relearning in Boltzmann machines. *Pages 282-317 of Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations. MIT Press*. 1986.

[50] H. Ma D. Zhou, C. Liu, M. Lyu,R. Michael and I. King. Recommender systems with social regularization. *Pages 287-296 of: Proceedings of the fourth ACM international conference on Web search and data mining, ACM*. 2012

[51] R. J. Mooney, L. Roy. Content-based book recommending using learning for text categorization. *Pages 195-204 of: Proceedings of the Fifth ACM Conference on Digital Libraries, ACM*. 2000

[52] G. Salton, M. McGill. Introduction to Modern Information Retrieval.*McGraw Hill*. 1983

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

[54] W. W. Cohen, R. E. Schapire, and Y. Singer. Learning to order things. *Pages 243-270 of: Journal of Artificial Intelligence Research*. 1999

[55] S. Amit. Modern Information Retrieval: A Brief Overview. *Pages 35-43 of: Bulletin of the IEEE Computer Society Technical Committee on Data Engineering 24 (4)*. 2001

[56] E. F. Krause. Taxicab Geometry. *Dover* . 1987

[57] W. Kim, O. Jeong and S. Lee. On social Web sites. *Pages 213-236 of: Information Systems (35)*. 2009

[58] S. Baluja, R. Seth, D. Sivakumar, Y. Jing, J. Yagnik, S. Kumar, D. Ravichandran and M. Aly. Video Suggestion and Discovery for Youtube: Taking Random Walks through the View Graph. *Pages 895-904 of: In WWW '08: Proceedings of the 17th International Conference on World Wide Web, ACM*. 2008

[59] J. L. Herlocker, J. A. Konstan, L. G. Terveen and J. T. Riedl. Evaluating Collaborative Filtering Recommender Systems. *Pages 5-53 of: ACM Transactions on Information Systems, 22(1)*. 2004

[60] G. Casella, E. I. George and I. Edward. Explaining the Gibbs sampler. *Pages 167-174 of: The American Statistician 46 (3)*. 1992

[61] E. Deza, M. M. Deza. *Pages 94-94 of: Encyclopedia of Distances, Springer*. 2009

[62] Yelp. *yelp.com*. Stated in January 2013.

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

[64] H. Ma, H. Yang, M.R. Lyu, and I. King. Sorec: social recommendation using probabilistic matrix factorization. *In Proc. of the 17th Conf. on Information and knowledge management*. 2008

[65] X. Amarin. Mining large streams of user data for personalized recommendations. *Pages 37-48 of: SIGKDD Explorations 14(2)*. 2012

[66] M. Jamali and M. Ester. Trustwalker: a random walk model for combining trust-based and item-based recommendation. *In Proc. of KDD '09*. 2009

[67] R. Salakhutdinov. Learning deep generative models. *Ph.d. Thesis University of Toronto, Toronto*. 2009

[68] T. Tran, D. Q. Phung, S. Venkatesh. Embedded Restricted Boltzmann Machines for fusion of mixed data types and applications in social measurements analysis. *Pages 1814-1821 of: FUSION 2012*. 2012

APPENDIX A

DATASET SCHEMES

A.1 SOCIAL GRAPH

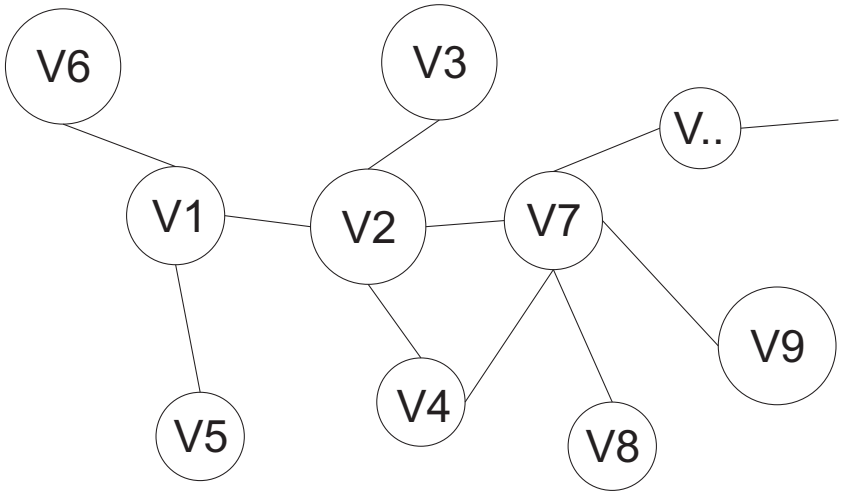


Figure A.1: Example Social Graphs of our Dataset

A.2 SQL SCHEMES

In order to test our algorithms, we need to use data set that are realistic representations of the production operation of a professional social network. These are full sql schemas that show the entire object relationships and collected data and They give us a wide data surface when deciding how to propose recommendation in a professional social network.

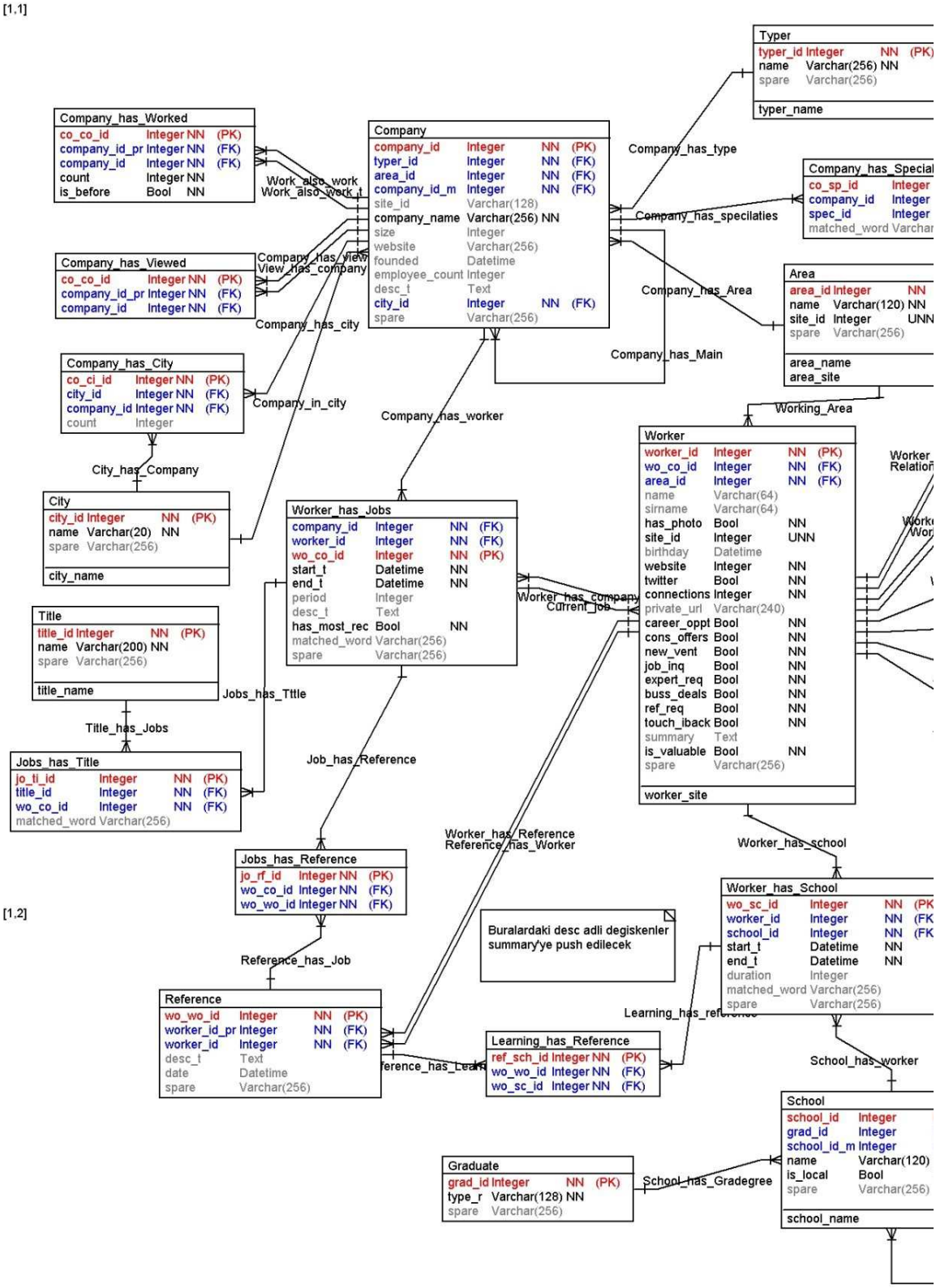


Figure A.2: Detailed Sql Schema of our Dataset Part-1

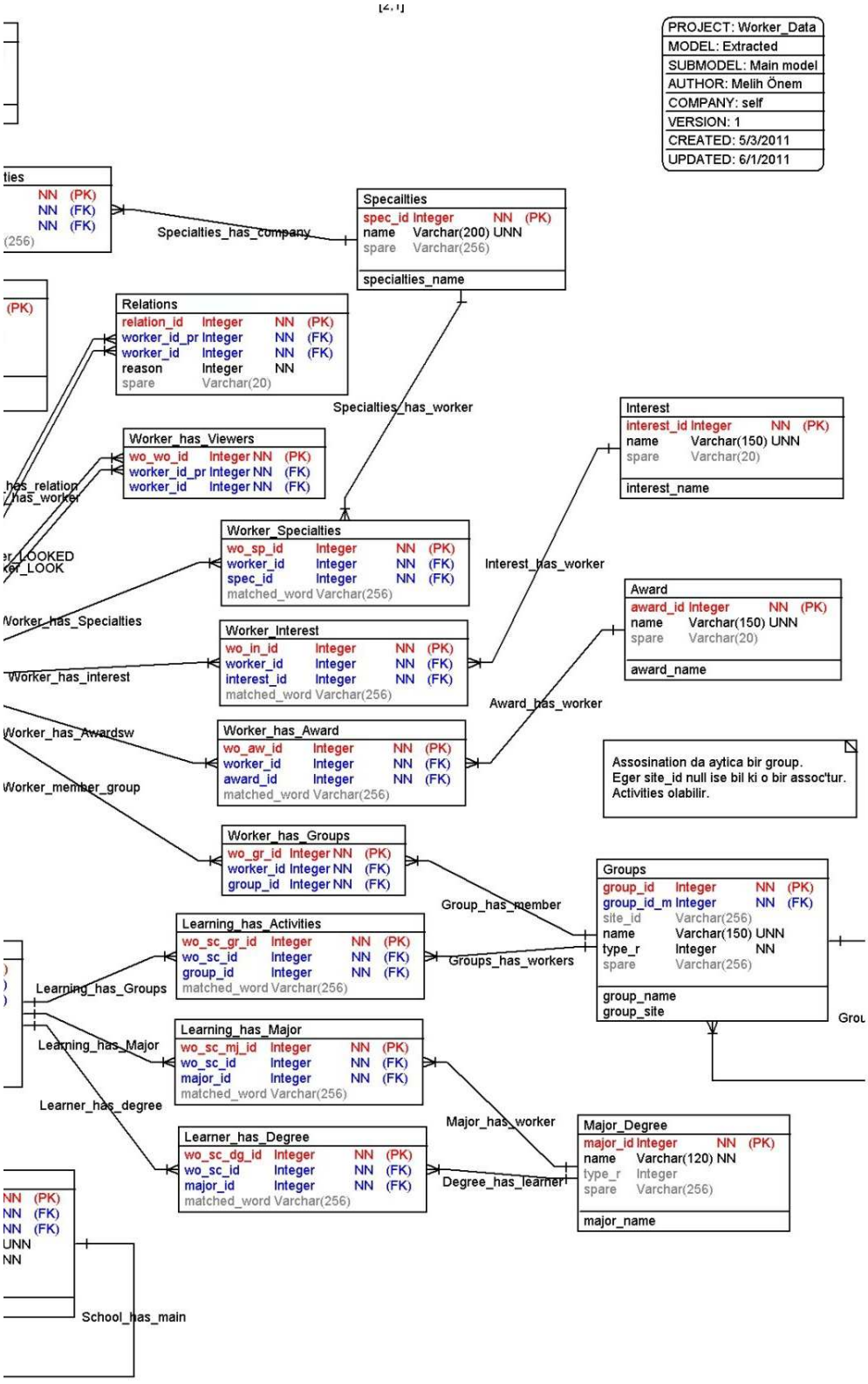


Figure A.3: Detailed Sql Schema of our Dataset Part-2

APPENDIX B

EVALUATION RESULTS

B.1 RESULTS OF MUTUAL FRIEND

B.1.1 COSINE DISTANCE AND IMPACT OF FILTERING

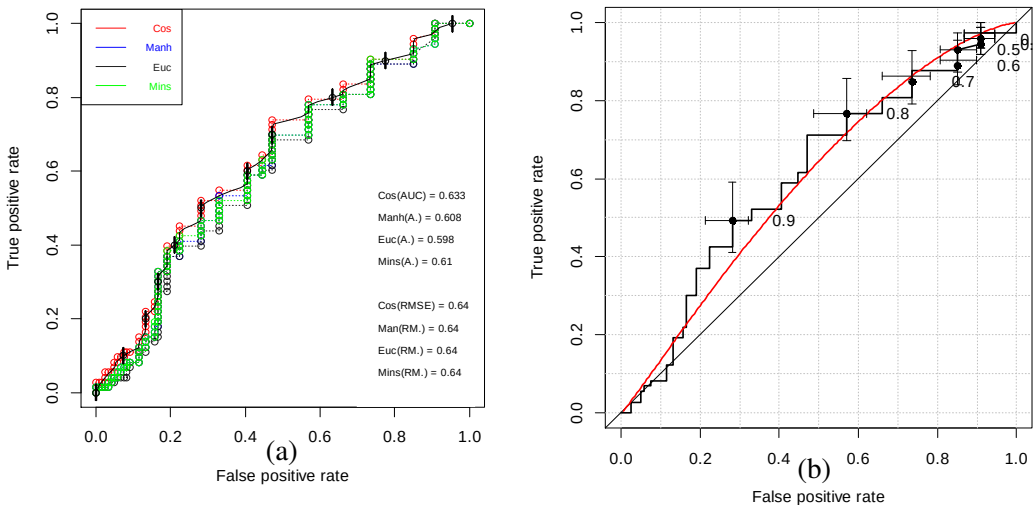


Figure B.1: (a) Comparison of the ROC Curves(with inclined form), AUC and RMSE predicted by WF social measure model against vector distance functions. Over 300 user tested with minimum 20 friendship sizes.(b) Average and cut-off points of four ROC Curves produced by evolution algorithm.

B.1.2 MANHATTAN DISTANCE AND IMPACT OF FILTERING

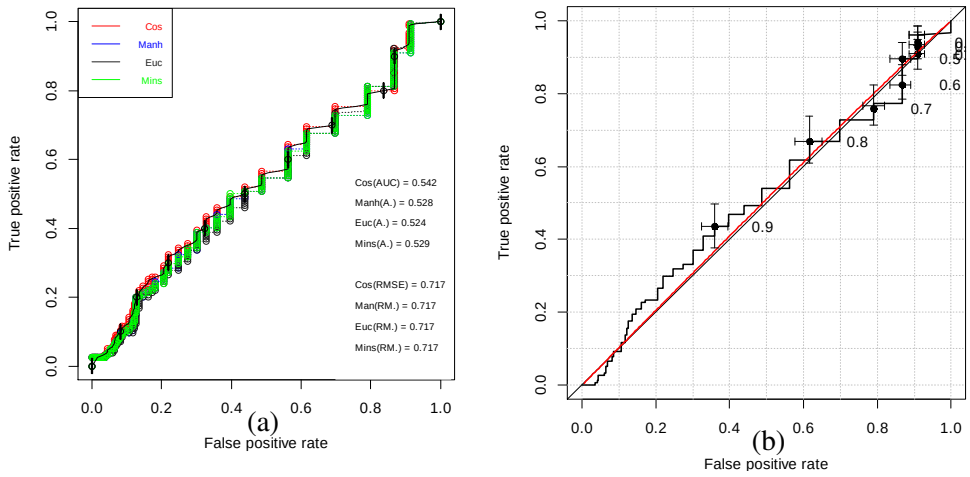


Figure B.2: (a) Comparison of the ROC Curves(with inclined form), AUC and RMSE predicted by MF social measure model against vector distance functions. Over 1000 user tested.(b) Average and cut-off points of four ROC Curves produced by evolution algorithm.

B.1.3 EUCLIDEAN DISTANCE AND IMPACT OF FILTERING

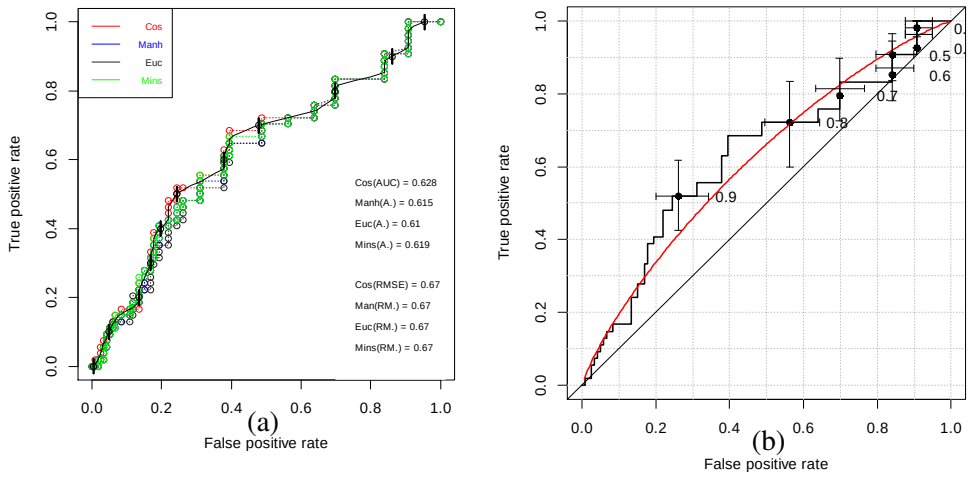


Figure B.3: (a) Comparison of the ROC Curves(with inclined form), AUC and RMSE predicted against vector distance functions. Over 175 user tested.(b) Average and cut-off points of four ROC Curves produced by evolution algorithm.

B.2 RESULTS OF WEIGHTED FRIEND

B.2.1 COSINE DISTANCE AND IMPACT OF FILTERING

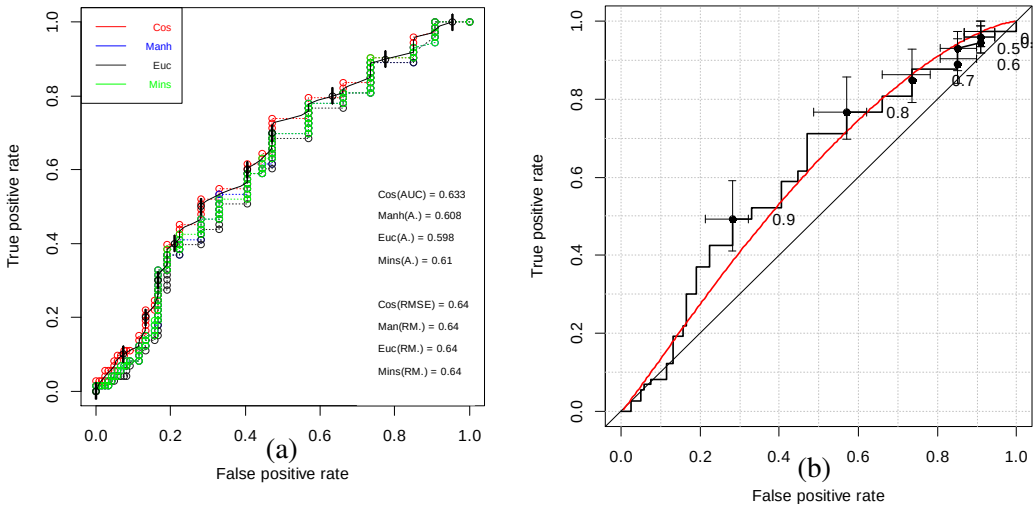


Figure B.4: (a) Comparison of the ROC Curves(with inclined form), AUC and RMSE predicted against vector distance functions. Over 300 user tested with minimum 20 friendship sizes.(b) Average and cut-off points of four ROC Curves produced by evolution algorithm.

B.2.2 MANHATTAN DISTANCE AND IMPACT OF FILTERING

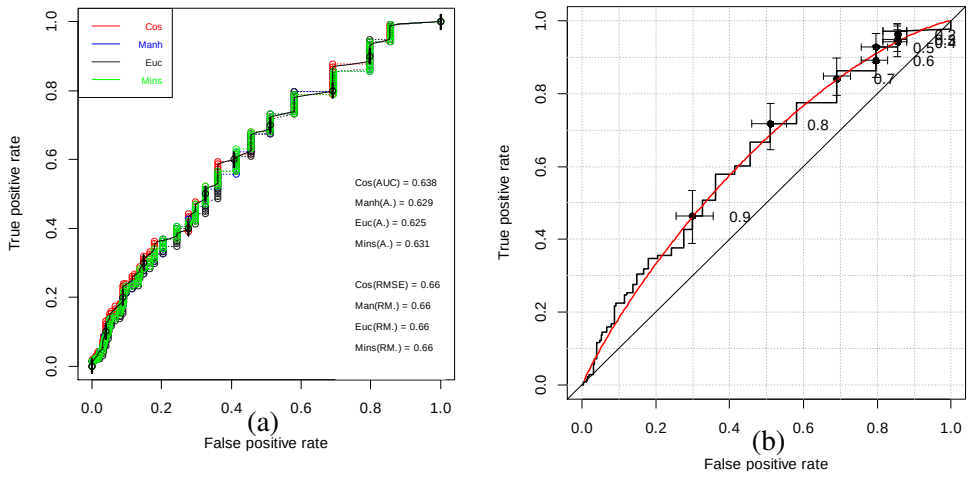


Figure B.5: (a) Comparison of the ROC Curves(with inclined form), AUC and RMSE predicted against vector distance functions. Over 1000 user tested.(b) Average and cut-off points of four ROC Curves produced by evolution algorithm.

B.2.3 EUCLIDEAN DISTANCE AND IMPACT OF FILTERING

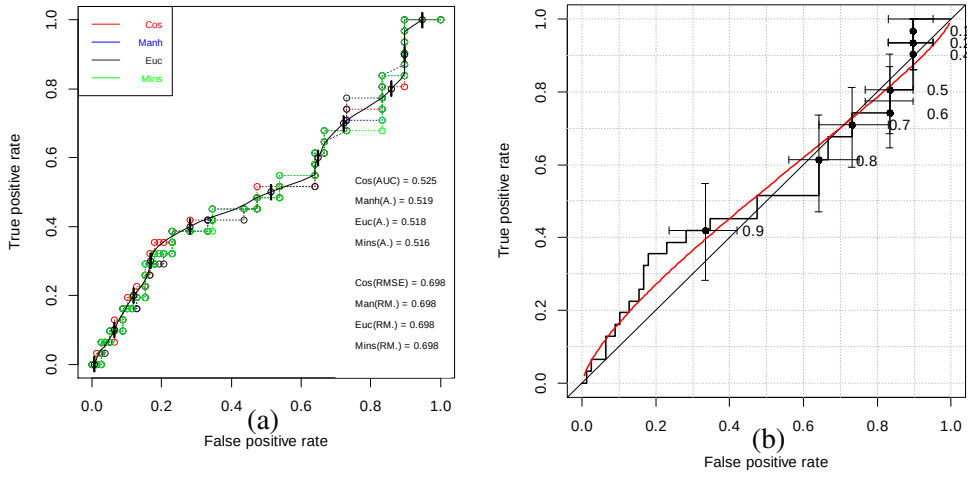


Figure B.6: (a) Comparison of the ROC Curves(with inclined form), AUC and RMSE predicted against vector distance functions. Over 175 user tested.(b) Average and cut-off points of four ROC Curves produced by evolution algorithm.

B.3 RESULTS OF FRIENDS' DISTANCE

B.3.1 COSINE DISTANCE AND IMPACT OF FILTERING

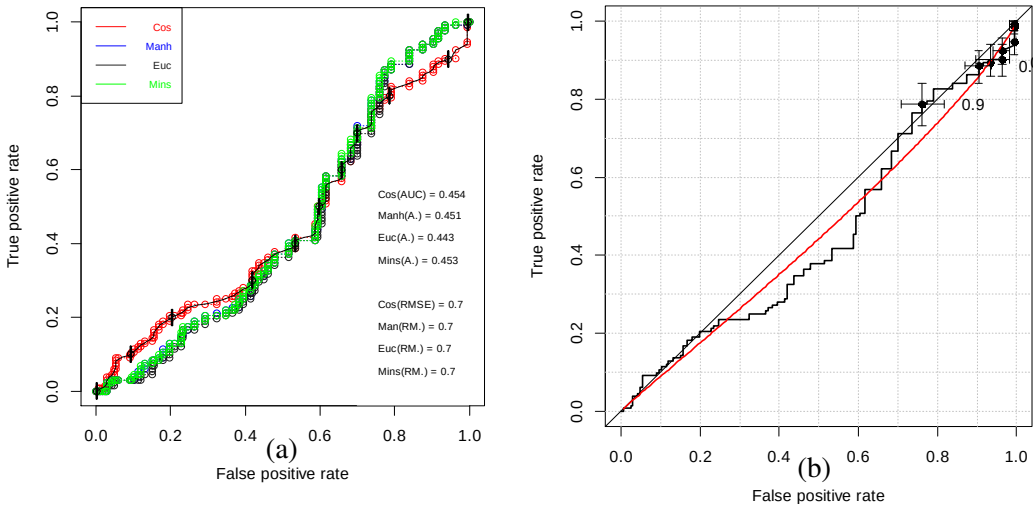


Figure B.7: (a) Comparison of the ROC Curves(with inclined form), AUC and RMSE predicted against vector distance functions. Over 300 user tested with minimum 20 friendship sizes.(b) Average and cut-off points of four ROC Curves produced by evolution algorithm.

B.3.2 MANHATTAN DISTANCE AND IMPACT OF FILTERING

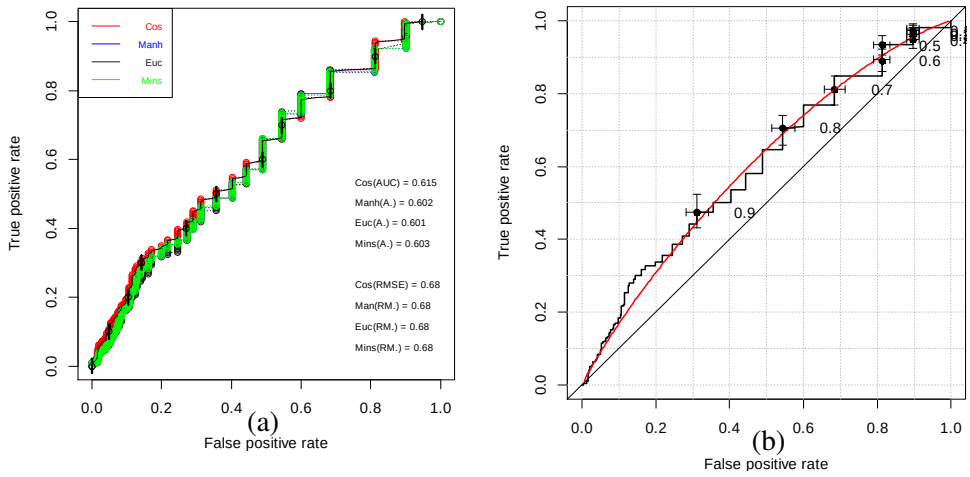


Figure B.8: (a) Comparison of the ROC Curves(with inclined form), AUC and RMSE predicted against vector distance functions. Over 1000 user tested.(b) Average and cut-off points of four ROC Curves produced by evolution algorithm.

B.3.3 EUCLIDEAN DISTANCE AND IMPACT OF FILTERING

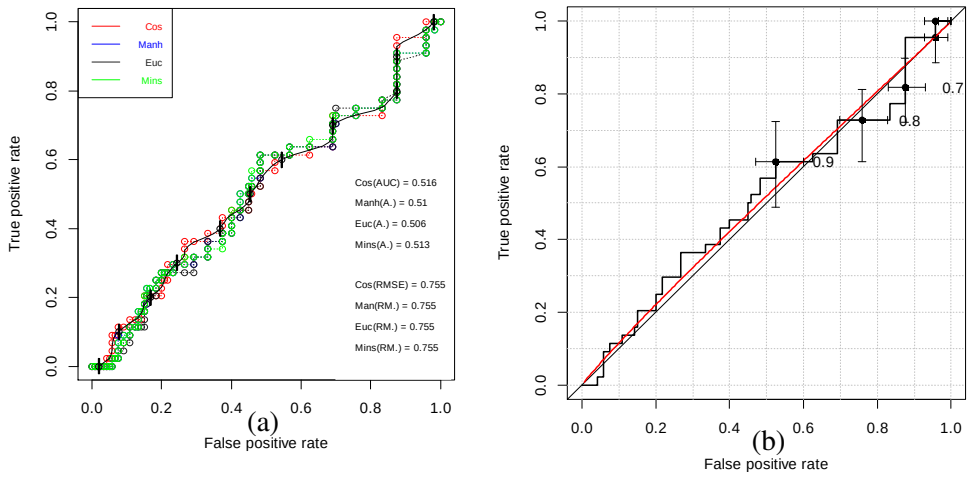


Figure B.9: (a) Comparison of the ROC Curves(with inclined form), AUC and RMSE predicted against vector distance functions. Over 175 user tested.(b) Average and cut-off points of four ROC Curves produced by evolution algorithm.

B.4 RESULTS OF FRIENDS' MUTUAL FRIENDS

B.4.1 COSINE DISTANCE AND IMPACT OF FILTERING

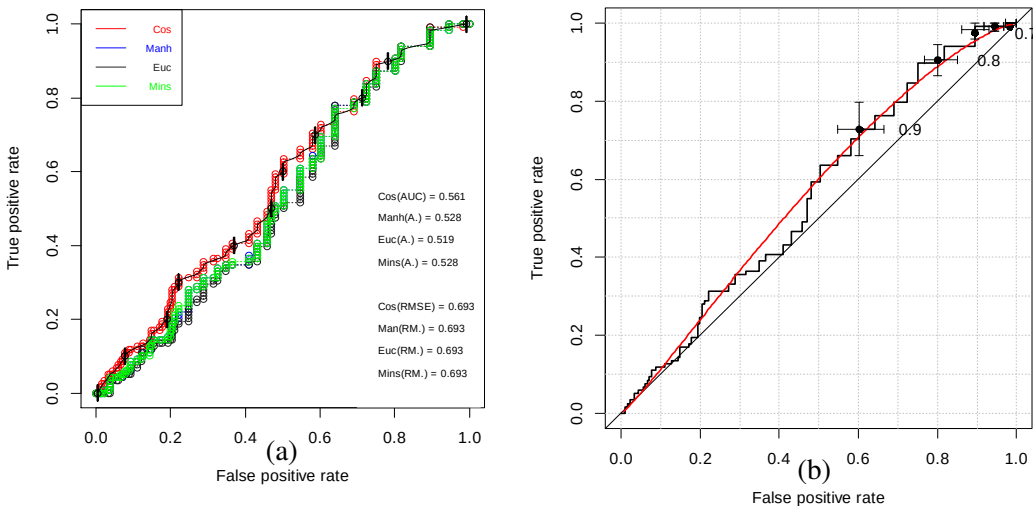


Figure B.10: (a) Comparison of the ROC Curves(with inclined form), AUC and RMSE predicted against vector distance functions. Over 300 user tested with minimum 20 friendship sizes.(b) Average and cut-off points of four ROC Curves produced by evolution algorithm.

B.4.2 MANHATTAN DISTANCE AND IMPACT OF FILTERING

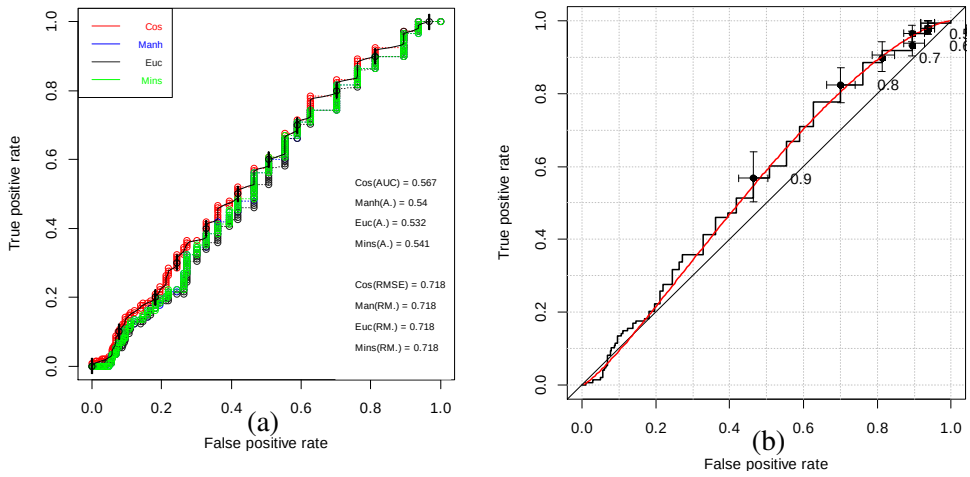


Figure B.11: (a) Comparison of the ROC Curves(with inclined form), AUC and RMSE predicted against vector distance functions. Over 1000 user tested.(b) Average and cut-off points of four ROC Curves produced by evolution algorithm.

B.4.3 EUCLIDEAN DISTANCE AND IMPACT OF FILTERING

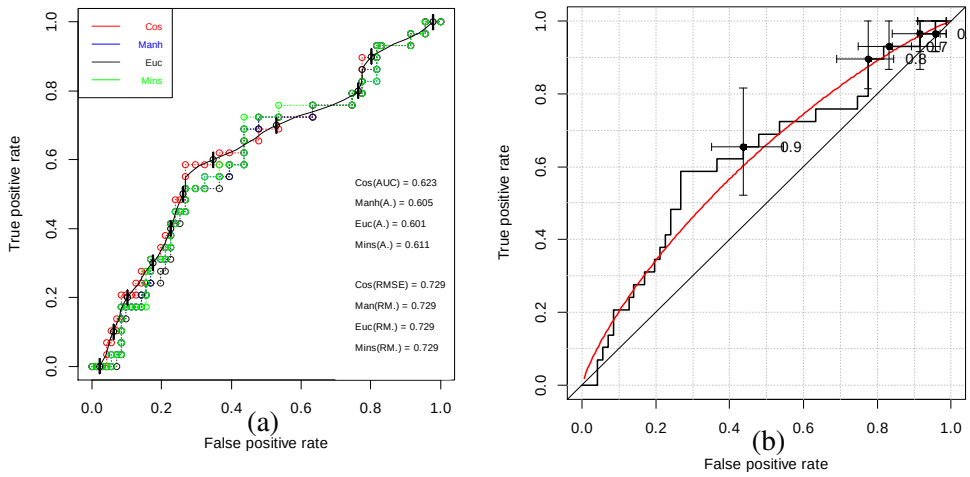


Figure B.12: (a) Comparison of the ROC Curves(with inclined form), AUC and RMSE predicted against vector distance functions. Over 175 user tested.(b) Average and cut-off points of four ROC Curves produced by evolution algorithm.

B.5 RESULTS OF BOLTZMANN MACHINES

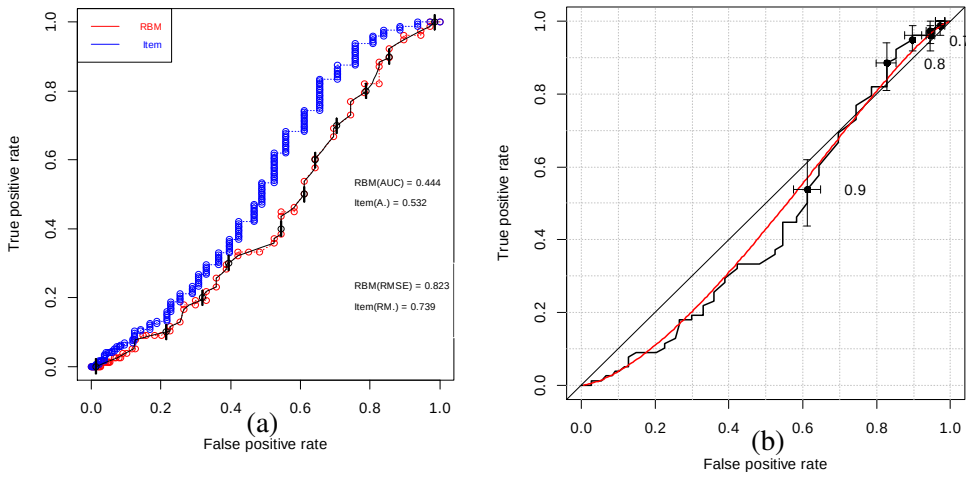


Figure B.13: (a) Comparison of the ROC Curves(with inclined form), AUC and RMSE predicted by FMD social measure model against RBM to Item to Item. Over 300 user tested.(b) Average and cut-off points of two ROC Curves produced by evolution algorithm.

B.5.1 IMPACT OF PRE-CLUSTERING

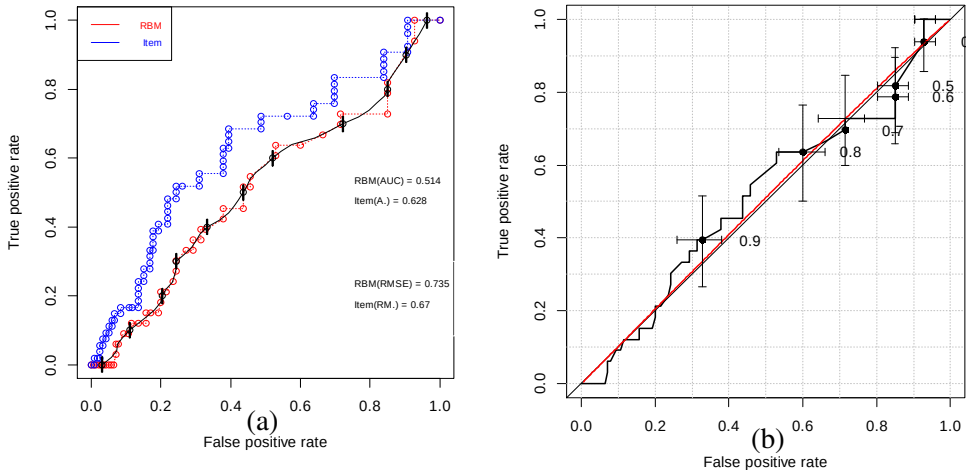


Figure B.14: (a) Comparison of the ROC Curves(with inclined form), AUC and RMSE predicted against cluster size downgrading 20 to 10. Over 300 user tested with minimum 20 friendship sizes.(b) Average and cut-off points of four ROC Curves produced by evolution algorithm.

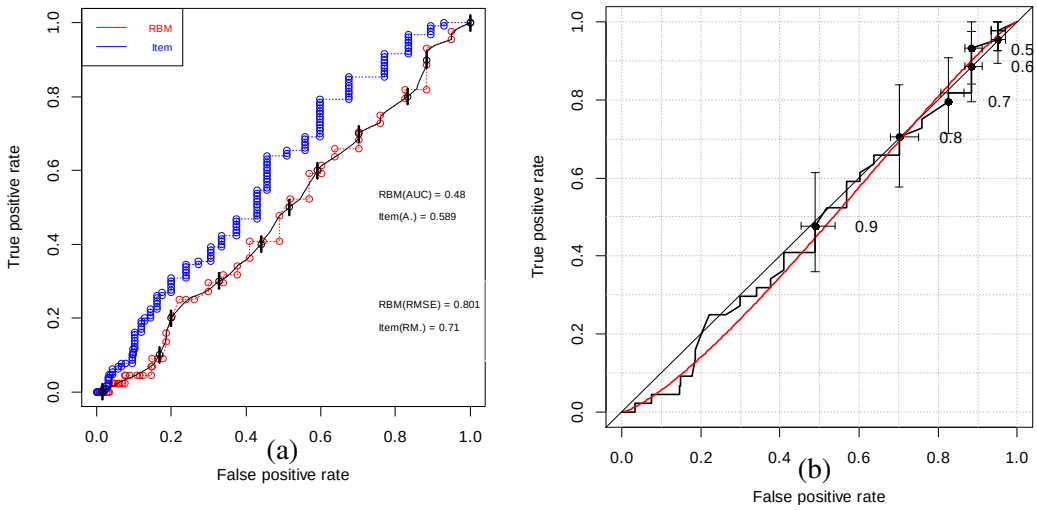


Figure B.15: (a) Comparison of the ROC Curves(with inclined form), AUC and RMSE predicted against vector distance functions cluster size downgrading 20 to 10. Over 300 user tested.(b) Average and cut-off points of four ROC Curves produced by evolution algorithm.