Incorporating User Reviews as Implicit Feedback for Improving Recommender Systems

by

Yasamin Heshmat Dehkordi

B.Sc., Isfahan University of Technology, 2011

A Thesis Submitted in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

Incorporating User Reviews as Implicit Feedback for Improving Recommender Systems

by

Yasamin Heshmat Dehkordi

B.Sc., Isfahan University of Technology, 2011

Supervisory Committee

Supervisory Committee

ABSTRACT

Recommendation systems have become extremely common in recent years due to the ubiquity of information across various applications. Online entertainment (e.g., Netflix), E-commerce (e.g., Amazon, Ebay) and publishing services such as Google News are all examples of services which use recommender systems. Recommendation systems are rapidly evolving in these years, but these methods have fallen short in coping with several emerging trends such as likes or votes on reviews. In this work we have proposed a new method based on collaborative filtering by considering other users' feedback on each review. To validate our approach we have used Yelp data set with more than 335,000 product and service category ratings and 70,817 real users. We present our results using comparative analysis with other well-known recommendation systems for particular categories of users and items.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to thank:

Dr. Sudhakar Ganti and Dr. Alex Thomo, for mentoring, support, patience and guidance through this research.

Dr. Issa Traore for his time and helpful comments.

My mom, for always being an inspiration in my life and for raising me with no fear of the future.

My dad, for being the symbol of patience, kindness and perseverance in life.

My brothers, for never leaving me alone and always guiding me to the best.

My husband, Nima for his endless support and love.

DEDICATION

I would like to dedicate this thesis to my family. If it wasn't for their love and patience, pursuing this path would have been impossible.

Chapter 1

Introduction

1.1 Problem Definition and Motivation

Internet has become one of the largest platforms for information production. To cope with this overload of information, effective retrieval and filtering techniques such as recommendation systems are being used. These systems discover user's unspecified information needs through multiple different approaches. They automatically suggest relevant information to a user without explicit input. In recent years recommendation systems have become a key component in various online services specially in E-Commerce (e.g.,Amazon, Ebay) and online entertainment (e.g., Netflix, Pandora). These systems usually suggest a list of items to users linking the user to the top predicted options of products or services.

Since the Netflix Prize competition, Collaborative Filtering (CF) [14] has become one of the most popular approaches for recommendation systems because of its simplicity and accuracy. Well known services such as Amazon, iTunes and Netflix are among the services that use collaborative filtering method for recommendation. Collaborative filtering depends on wisdom of the crowd. These methods produce user specific recommendations for items based on ratings or usage patterns (e.g, purchases, browsing history, etc.). These recommendations are computed without the need for external information about items or users. For example, the neighborhood method [2] is one of the approaches in collaborative filtering that finds other users (neighbors) with similar tastes based on their preferences. The preferences expressed by the neighbors can then be aggregated to recommend items to the target user.

For using and comprehending implicit and explicit data received from users, several aspects and characteristics can be extracted about the user [18, 31]. As an example there are methods that model *how* users make their assessment for different purposes, such as suggesting new products they might enjoy [19] or identifying other users who may share similar opinions [30]. In general reviews give us an insight to the user's method of thinking and rating. In some applications like Yelp services there are tags such as "vote" which other users can vote if the review was either "funny", "useful" or "cool". Considering these extra details in reviews can help improving the recommendation system.

To the best of our knowledge the impact of implicit feedback from other users on reviews has not been addressed. Therefore, we have proposed an algorithm to improve recommender system performance by taking into account other users' opinions on reviews.

1.2 Research Contribution

Our proposed algorithm in this thesis is based on the user-based collaborative filtering techniques [2] and Koren Bell's algorithm [13, 14]. In this method not only the rates given in a review are used but also implicit feedback from other users is taken into consideration by including the impact of useful votes on each review. To validate our approach with real data, we used Yelp academic data set [11] which consists of 335,022 reviews and 70,817 real users. The outcome of our suggested algorithm shows a Root Mean Square Error (RMSE) improvement of 9.46% compared to the classical collaborative filtering. This is where even a 0.5% improvement can make a big difference in these systems. Further, a fine grained experimental analysis is conducted by categorizing users and items. The categorization is based on the number of user reviews and the deviation of their ratings from the average rating to see the effect of system suggestions for each category of users and items. Unlike methods that have a high threshold for the number of reviews of users [7], our approach has low error rate on test sets containing users with just more than 5 reviews. Finally, our results are presented by using different comparative analysis that involves other well known recommendation models.

1.3 Thesis Outline

Sections of the thesis are organized as follows.

Chapter 2 presents related background and literature review along with the research problem and its impact.

Chapter 3 presents an overview of our recommendation system. We also introduce our algorithm and its components. The data set used in this research is described in detail.

Chapter 4 presents the test set used for evaluation and its results. Furthermore, in this chapter results are compared with well-known methods by several indicators and performance metrics.

Chapter 5 concludes this research and enumerates avenues of future work.

Chapter 2

Related Work and Background

In this chapter we review main methods and algorithms used for recommender systems. The purpose of this chapter is to discuss the main concepts and methods related to the research and to create the perspective of the problem at hand and the motivation of this study.

2.1 Recommender Systems

Recommender Systems (RSs) are software tools and techniques that provide suggestions for items to users. The recommendation and suggestions made are related to various decision-making processes. In these systems "item" is a term that is generally used to indicate what the system suggests to users. Since every genre of item has its own characteristics (e.g. music, restaurants), RSs normally focus on a specific type of item. In other words, the core techniques used for generating the recommendations are all customized to provide useful and effective suggestions for that specific type of item [24]. Movies to watch, items to buy, music to listen to and books to read can be mentioned as some items that recommenders suggest to users. RSs are mainly used when individuals lack sufficient personal experience or competence to evaluate the potentially overwhelming number of alternative items that a service offers [28]. Recommendations are personalized; i.e., they are usually made for the purpose to answer what each individual user is seeking for. Each individual user or user group receives diverse suggestions. As an example of a personalized recommendation system consider Netflix where each user gets suggestions for movies based on other movies that he/she has rated or browsed into.

Research in the field of recommendation system has been started from mid-1990s. In the past decade this research area has gained more popularity because of its main usage in everyday life and its application in websites such as Youtube and Amazon [15]. The following sections review different techniques and classifications of the recommender systems.

2.2 Recommendation Methods

Most of the current recommendation techniques use a single criterion such as a single numerical rating to represent an item's utility to a user in a two dimensional (2D) Users Items space such as table 2.1 [1]. The previous ratings of users are used for predicting the unrated items by the user. The most common way of recommending is to suggest the top N highest predicted rate to the user. Table 2.1 shows a basic foundation of a recommender system. In table 2.1 the (user, item) pairs that the user has not yet rated are marked with a question mark "?". Different methods can be used to predict the unrated items by a user which will be discussed later in this chapter. After predicting the ratings the system will recommend N recommendations from the top predicted ratings.

Table 2.1: User-Item rating matrix

	$item_1$	$item_2$	$item_3$	$item_4$
$user_1$	?	4	3	4
$user_2$	2	5	?	5
$user_3$	3	?	3	4
$user_4$	2	3	3	?

For better understanding of the recommendation systems, consider the users and items from table 2.1. $user_1$ assigns a single-criterion rating of 3 (out of 5) for $item_3$; in other words, $R(user_1, item_3) = 3$. Once the initial ratings are specified, a recommender system tries to estimate the rating function R for the unrated (user, item) pairs.

$$R : Users \times Items \to R_0 \tag{2.1}$$

R_0 is a set of values that a predicted rating can take which is usually represented by a totally ordered set, for example, integers or real numbers within a certain range. Once the system estimates function R, it can recommend the highest rated item or a set of N highest-rated items for each user. Let u be a utility function which indicates the usefulness of item s to user c.

$$u : C \times S \to R \tag{2.2}$$

A typical recommender system goal is to correctly estimate the ratings of unrated items. The second main goal is usually to find items with a maximum utility for each user [1].

There are many methods for predicting unknown ratings that are classified into the following categories [2]:

- **Content-based recommendations:** Recommended items in this method are the items similar to user's previous preferred and chosen items.
- **Collaborative recommendations:** This method recommends items to the user based on items which similar users have chosen in the past.
- **Hybrid approaches:** Combination of collaborative and content-based methods results in Hybrid method. This method is used for overcoming limitations of the previous two recommender methods.

2.2.1 Content-based Methods

In Content-based methods, recommending item s to user c is based on the similarity between the content of s and the previously rated items by user c. For example if a user named John has favored action movies and has rated them highly in the past, action movies will have higher similarity rate for John. Based on the high similarity, the system will recommend him movies from action genre. Content-based systems need to have user profiles and methods for representing the items as well as comparison strategies between these two. The recommendation process in content-based method is performed in three steps that can be summarized as below [24]:

- **Content Analyzer:** The main responsibility of this module is to represent the content of items (texts, web pages, product description etc.) into a structured information suitable for the next processing steps. Feature extraction techniques are used in order to shift item representation from the original information space to the target one. As an example, text would be represented as keyword vectors.

- **Profile Learner:** In this module, system will collect user's preferences and generalize the data based on machine learning techniques to construct a profile for each user. These techniques are able to infer a model of user interests from items liked, visited or disliked in the past. The learning technique combines vectors of positive and negative examples into a prototype vector representing the user profile.

- **Filtering Component:** This module uses the user profile to suggest relevant items by matching the profile representation against items to be recommended. The result is a similarity matrix which reveals a ranked list of items that are interesting to the user.

The formulation of these systems is as follow: Let $Profile(c)$ be the user c profile representing the likes and dislikes of the user. This profile is a set of keywords extracted from the user actions with analysis techniques. Let $Content(s)$ be the profile of the item s representing the features and attributes of the item by keywords. The utility function in this system is defined as:

$$u(c, s) = similarity\ score(Profile(c), Content(s)) \tag{2.3}$$

The profiles of users and items are the information we have which is in the form of keywords. It is essential to know which of these keywords are mostly important for characterizing the items and the users. That is why there is a need to know the "importance" of these keywords. Assume that each user profile or item content can be represented by a document d_j. The "informativeness" or the "importance" of the word k_j in document d_j is determined with some weighting measure w_{ij} which can be defined in several different ways. One of the popular measures for specifying keyword weight is the term *Frequency/Inverse Document Frequency (TF-IDF)* measure [2, 25]. Assume N is the number of documents that can be recommended to users, the keyword k_i appears in n_i of them and $f_{i,j}$ is the number of times keyword k_i appears in document d_j . Then the frequency or the the normalized frequency (TF) of k_i in document d_j is defined as:

$$TF_{i,j} = \frac{f_{i,j}}{max_z f_{z,j}} \tag{2.4}$$

where maximum is obtained over the frequencies $f_{z,j}$ of all keywords k_z that appear in d_j.

The IDF or *Inverse Document Frequency* in this method can be defined as follows: Let N be the total number of documents and n_i the number of documents containing keyword k_i. IDF of keyword i is :

$$IDF_i = \log \frac{N}{n_i} \tag{2.5}$$

Finally the TF-IDF weight or importance and informativeness of keyword k_i in document d_j is defined as:

$$w_{i,j} = TF_{i,j} \times IDF_i \tag{2.6}$$

Using this measurement we can characterize the content of the document d_j by using important words as $Content(d_j) = (w_{1j}, \ldots, w_{kj})$. The content of both user profile ($Profile(c)$) and item content ($Content(s)$) are represented as TF-IDF vectors $\vec{w_c}$ and $\vec{w_s}$ of keyword weights. After calculating these vectors from the user and item profiles, the similarity between the items need to be measured in order to recommend items similar to what he/she has liked in the past in the profile. By calculating the similarity, one can determine the items with highest similarities that are among the best recommendations for the user based on his/her preferences on the profile. By using cosine similarity [3], the utility function is defined as:

$$u(c,s) = \cos(\vec{w_c}, \vec{w_s}) = \frac{\vec{w_c}.\vec{w_s}}{\|\vec{w_c}\|_2 \times \|\vec{w_s}\|_2} \tag{2.7}$$

where $||.||_2$ is the Euclidean norm of the vector. Besides the traditional method introduced above other techniques for content-based recommendation have also been used such as Bayesian classifiers [2],[20],[21] and various machine learning techniques, including clustering, decision trees and artificial neural networks. These techniques differ from the information retrieval-based approaches in a sense that they calculate utility predictions based on a model learned from the data using statistical learning and machine learning techniques.

As of any other methods, there are limitations associated with content-based methods. The main limitation is that the content of items must be in a form that can be automatically parsed by a computer such as text, otherwise the features need to be assigned to items manually. The domains that the automatic feature extraction cannot be much of a use are usually the ones associated with audio and video. Another limitation is that content-based system cannot distinguish and differ between two items with the same set of features. Therefore, since text based documents are only represented by their most important keywords, a well-written book and a badly written one will be treated the same if they use the same keywords [29]. Another limitation is over-specialization. Over-specialization is when users are only given recommendations related or similar to the items they liked in the past and not in other fields and categories especially for the new-users. A new-user is a problem area in these systems due to lack of user profile; and the system is not able to understand the user's characterization and preferences to recommend relative items. In these systems a user has to have enough activity to have a profile. Therefore, the system can only understand the user only if there are enough ratings.

2.2.2 Collaborative Recommendation Methods

Collaborative recommender systems (also known as *collaborative filtering systems*) are among the most popular recommendation systems that have overcome most of the limitations of the content-based methods. Unlike the content-based recommendation methods, collaborative approaches try to predict the utility of items for a particular user. These predictions are based on the items previously rated by *other* users rather than the content of items previously rated by himself/herself. In other words, the system first finds the users with similar interests to the target user and then recommends the items highly rated by the similar users. Recommending item s for user c is based on the utilities $u(c_j, s)$ assigned to item s by user $c_j \in C$ where c_j are the users *similar* to user c [2]. The main idea here is that the rating of user u for a new item i is likely to be similar to that of another user v when u and v have rated other items in a similar way. In contrast with content-based method, items with unavailable content or with content difficult to obtain can still be recommended to users through the feedback of other users in collaborative filtering method.

Furthermore, collaborative recommendations are based on the quality of items as evaluated by users, instead of relying on the content that may be a bad indicator of quality. Collaborative filtering systems have the advantage of recommending items with very dissimilar content, as long as other users have already shown interest for these different items. This characteristic of the system overcomes the problem of over-specialization in content-based method. Algorithms for collaborative method can be grouped in two general classes of *neighborhood* (also know as *memory-based* and *heuristic-based* methods) and *model based* methods [5].

Memory-based Collaborative Filtering Method

In this system the user-item ratings are directly used to predict ratings for new items. This can be done in two ways known as user-based or item-based recommendation. For instance, GroupLens [12], Bellcore video [10] and Ringo [29] can be named as some of the well known user-based systems. Neighborhood or memory-based algorithms calculate rating predictions by considering previously rated items by the users. The value of the unknown rating $r_{c,s}$ for user c and item s is calculated as an aggregate of the ratings from the most similar users for the targeted item s [2].

$$r_{c,s} = \underset{c' \in \hat{C}}{aggr}\, r_{c',s} \tag{2.8}$$

Where $\hat{C}$ is a group of N users that are most similar to user c who have rated item s. The simplest method for considering all the ratings of the similar users is to have an average of their ratings:

$$r_{c,s} = 1/N \sum_{c' \in \hat{C}} r_{c',s} \tag{2.9}$$

However, the most popular method is to have a weighted sum with similarities being the weights in the formula:

$$r_{c,s} = k \sum_{c' \in \hat{C}} sim(c, c') \times r_{c',s} \tag{2.10}$$

k in the above formula is a normalizing factor that is usually calculated as below:

$$k = \frac{1}{\sum_{c' \in \hat{C}} |sim(c, c')|} \tag{2.11}$$

In equation (2.10) $sim(c, c')$ is the similarity of user c to c' and vice versa. We know that each of the user has his/her own standards for ratings, which the system above does not consider. To fix the problem, deviations from the average ratings of users are considered by using adjusted weighted sum [23].

$$r_{c,s} = \overline{r}_c + k \sum_{c' \in \hat{C}} sim(c, c') \times (r_{c',s} - \overline{r}_{c'}) \tag{2.12}$$

and $\overline{r}_{c'}$ is defined as:

$$\overline{r}_{c'} = (1/|S_{c'}|) \sum_{s \in S_{c'}} r_{c',s} \quad , \quad S_{c'} = \{s \in S | r_{c',s} \neq \emptyset\} \tag{2.13}$$

As seen in aforementioned formulas, similarity is one of the main calculations. The similarity itself can be computed by different methods between two users. The similarity of two users is calculated by the ratings of items co-rated by both users. The similarity between two items is based on the ratings given by users to these items. The two most popular approaches for calculating similarity are:

- **Cosine-based:** This method considers the items rated by users u and v as two vectors in an *n-dimensional* space. Where n is the number of all items co-rated by both users u and v. The set of co-rated items can be defined as [5, 27]:

$$S_{uv} = \{s \in S | r_{u,s} \neq \emptyset \ \& \ r_{v,s} \neq \emptyset\} \tag{2.14}$$

By considering $n = |S_{uv}|$, the similarity between the vectors $\vec{u}$ and $\vec{v}$ can be computed with the cosine of the angle between these two:

$$sim(u, v) = cos(\vec{u}, \vec{v}) = \frac{\vec{u}.\vec{v}}{\|\vec{u}\|_2 \times \|\vec{v}\|_2} \tag{2.15}$$

Note that the same cosine measure is used in both the content-based and the collaborative approaches. The difference is that in content-based recommender systems, cosine similarity is used to measure the similarity between vectors of TF-IDF weights, whereas, in collaborative systems, it measures the similarity between vectors of the actual user-specified ratings.

- **Correlation-based:** This method uses the Pearson Correlation Coefficient for measuring the similarity between users u and v [2, 23, 29].

$$sim(u,v) = \frac{\sum\limits_{s \in S_{uv}} (r_{u,s} - \overline{r}_u)(r_{v,s} - \overline{r}_v)}{\sqrt{\sum\limits_{s \in S_{uv}} (r_{u,s} - \overline{r}_u)^2 \sum\limits_{s \in S_{uv}} (r_{v,s} - \overline{r}_v)^2}} \tag{2.16}$$

Many performance-improving modifications have been proposed as extensions to compute similarity with correlation-based and cosine-based techniques such as *default voting, inverse user frequency* and *case amplification* [5]. It was observed that neighborhood CF methods (or memory-based methods) overcome some limitations mentioned for content-based systems. For instance, these methods can recommend items that have contents difficult to obtain or have no content at all since these recommenders suggest using the feedback of other users.

These user-based recommender systems have their own limitations. One of the main limitations is that the computation needed for obtaining real-time similarity between millions of neighbors can be impossible. For overcoming this problem usage of clusters and computing the similarity between these clusters is suggested. Clustering leads to less computation and results in a faster process, but the downside is lower quality of the recommendations [2].

Model-based Collaborative Filtering Methods

Unlike the memory-based method, model-based collaborative filtering method does not consider all the users. Instead it uses a *collection* of users to *learn a model* that has advantage of learning and recognizing complex patterns based on the training data. These learned models will be used to predict the ratings at a later stage. As a good example, consider the model that uses probabilistic approach to collaborative filtering with the estimation of the unknown ratings as below [5, 2]:

$$r_{c,s} = E(r_{c,s}) = \sum_{i=0}^{n} i \times Pr(r_{c,s} = i | r_{c,s'}, s' \in S_c) \tag{2.17}$$

In (2.17) the probability expression is the probability that user c will give a particular rating to item s given that user's ratings of the previously rated items. To estimate the probability mentioned as Pr there has been two methods proposed by [5] : *clustered models* and *Bayesian networks*

- *Clustered models:* In this method similar users are clustered into classes and are given class membership. The number of classes and all the parameters are learned from the data.

- *Bayesian network:* In this method the structure of network and conditional probablities are learned from the data. Each item in this model is represented as a node in the Bayesian network. The state of each node corresponds to the possible rating values for each item. The limitation of this method is that a user cannot be assigned to different categories at once.

The above model was just an example of several other model-based approaches.
The main difference between the model-based method and memory-based method in collaborative filtering is that model-based method computes ratings for its prediction based on an accurate model learned from the data using machine learning and statistical methods. This is unlike the memory-based method which considers ad-hoc heuristic methods for rating prediction.

Collaborative Filtering (CF) Limitations

As of any other system CF has its own limitations. Main limitations of these systems are the new item problem, the new user problem and the sparsity of the data. The new user problem is similar to the situation in content-based method. Enough recommendation has to be at hand to have accurate prediction for unknown ratings. The solution to this problem is to use the hybrid recommendation approach. The new item problem is due to the system relying upon the preferences of users, but when a new item is added to the system no recommendation can be made until the item is rated by some of the users.
In all the recommender systems the number of unknown ratings that need to be predicted is always more than the known ratings. The main quality character of the CF recommender is its ability to give correct recommendations with small number of examples. Besides the need for many recommendations for items, there is always some people that are not similar to anyone. The system thus will make inaccurate recommendations for unique users [4]. A solution to this problem is considering a

user profile for recommendation information such as gender, age, education and etc. These are extensions made to the main collaborative filtering method named "demographic filtering" [22]. Another solution for the sparsity of the rating matrix is the usage of Singular Value Decomposition (SVD) which is a popular method for matrix factorization [2].

2.2.3 Hybrid Reommendation Method

Hybrid methods are combination of collaborative filtering and content-based methods. Hybrid method is a solution to the limitations of collaborative filtering and content-based methods. The approaches of combining content-based with collaborative filtering methods for creating hybrid systems can be summarized in the categories below [2]:

- Combination of prediction results from the two systems which have been implemented separately.
- Incorporating some content-based characteristics into a collaborative approach.
- Incorporating some collaborative characteristics into a content-based approach.
- Producing a general model that unifies and incorporates both content-based and collaborative characteristics.

2.2.4 Advances in Collaborative Filtering

The collaborative filtering method creates recommendations of items based on the item usage or pattern of ratings without the use of external information about the items or the users. The Netflix Prize competition which began in October 2006 caused a big step in the field of collaborative filtering. For the first time through this competition, research community gained access to a large-scale, industrial strength data set of 100 million movie ratings attracting thousands of scientists to the field. On September 2009, the grand prize of 1 million dollars was given to the "BellKor's Pragmatic Chaos" team which succeeded in raising Netflix's own algorithm for predicting ratings by 10.06 % [14].

For recommendation, collaborative filtering systems need to relate the items and users. The two main techniques of advanced CF are: *neighborhood* approach and *latent factor models*. Neighborhood method - as the name indicates - is based on relationships between items or between users. Item-based method is relative to the ratings of similar items by the target user. As one of the main Latent factor models, matrix factorization (e.g. SVD) creates a different approach by transforming both items and users to the same latent factor space. In the following subsection the neighborhood method which has been used in this research will be discussed in more details.

Baseline Predictors

For better understanding of the user preferences in the collaborative filtering systems there is a need to know the user-item interactions. Much of the rating values observed are usually independent of user and item interaction. As we know every person has their own scale for ranking items. For example, someone might think 3 out of 5 is a good rating while some other may think that this is a low rating to give to an item. Same applies to the items, some items have higher ranking say a good movie such as Brave Heart than average movies. For overcoming these effects that are independent from the interactions between items and users, we use a method called *Baseline Predictors (BP)* or in short form *biases.* This method considers a signal that is based on the user-item interaction as follows:

$$b_{ui} = \mu + b_u + b_i \tag{2.18}$$

Where b_u indicates the deviation of user u from an average user and b_i indicates the deviation of item i from average. For example, consider that we want to compute baseline predictor for a TV series named House M.D. by a user named Ariana. The average rating over all TV series is 3.5 out of 5. House M. D. is a well known TV drama series which is above average by 0.5 stars. Also Ariana is a critical user who tends to rate 0.2 stars lower than average. Therefore the baseline predictor for House M.D. for Ariana is 3.8:

$$b_{ui} = \mu + b_u + b_i \rightarrow b_{ui} = 3.5 + (-0.2) + 0.5$$

b_u and b_i can be computed by solving the least squares problem:

$$\min_{b^*} \sum_{(u,i)\in k} (r_{ui} - \mu - b_u - b_i)^2 + \lambda_1 (\sum_u b_u^2 + \sum_i b_i^2) \tag{2.19}$$

The first term in this optimization problem is to find b_u and b_i in such a way that fits the given ratings. The next term in this equation is a regularizing term

$(\lambda_1(\sum_u b_u^2 + \sum_i b_i^2))$ which avoids over fitting by penalizing the magnitudes of the parameters by λ_1. This optimization problem can be solved by stochastic gradient descent method [14]. An easier method but with less accuracy is to decouple the calculation of the b_i's from b_u's. For this purpose we have to first compute b_i for each item i as:

$$b_i = \frac{\sum_{u \in R(i)}(r_{ui} - \mu)}{\lambda_2 + |R(i)|} \tag{2.20}$$

Where $R(i)$ are the users who have rated item i. For each user u then we can define b_u as :

$$b_u = \frac{\sum_{i \in R(u)}(r_{ui} - \mu - b_i)}{\lambda_3 + |R(u)|} \tag{2.21}$$

In the equation above $R(u)$ indicates the items rated by user u. The value of λ_2 and λ_3 are computed by the method called cross validation. The values used for the Netflix data set were $\lambda_2 = 25$ and $\lambda_3 = 10$ [14].

Implicit Feedback

Browsing history or the items that user tends to choose to rate are considered as implicit feedback. This feedback gives more information about the user preferences without requiring them to provide explicit information. The data set does not only tell us the rating values, but also which item users rate, regardless of the value of the rating. In other words, a user implicitly tells us about his/her preferences by choosing to rate an item. For a data set such as the Yelp data set, we can consider the voting as an implicit feedback. These votes can tell about the user preferences and also the amount of trust that we can have on the review without requiring them to explicitly provide information.

Neighborhood Models

As mentioned in the traditional methods of collaborative filtering, neighborhood method (memory based method) is one of the two main methods in this category. In advanced collaborative filtering a bit of a twist has been added which will be discussed further.

Item-item based method recommends items similar to user's past choice of items [14]. In item-item based method a rating is estimated using known ratings made by the *same user* on *similar items*. However, user-user based method computes unknown ratings based on ratings of similar users. The neighborhood approach is mainly used for item-item based method rather than the user-user based method but it can be extended to be used in these systems as well. Latent factor model provides more accurate results than the neighborhood methods. However, the commercial usages of neighborhood model is more common such as in Amazon [16].

For a formal formulation, consider that we need to predict the unknown rating $\hat{r}_{ui}$ by using the similarity measurements. The following method is a popular approach to neighborhood method. In this method the goal is to predict r_{ui} which is the unknown rating by user u for item i. Using the similarity measure, the k most similar items to i rated by u are identified. This set of k neighbors is denoted by $S^k(i;u)$. After this process the value of $\hat{r}_{ui}$ can be computed by formula below [14]:

$$\hat{r}_{ui} = b_{ui} + \frac{\sum\limits_{j \in S^k(i;u)} S_{ij}(r_{uj} - b_{uj})}{\sum\limits_{j \in S^k_{(i;u)}} S_{ij}} \tag{2.22}$$

Where $\hat{r}_{ui}$ is the final estimation of the rating by user u for item i and S_{ij} is the similarity measure for item i to j.

This standard neighborhood method has some issues such as [14]:

- The similarity function that is used in these system varies. In other words, different Collaborative Filtering algorithms use different similarity measures. For instance, consider a particular item that was predicted by a subset of the neighbors. In that case, we would want the predictive subset to receive all the weights, but that is impossible for methods with bounded similarity scores like the Pearson Correlation Coefficient.

- Interactions between neighbors are not considered in these methods. In other words, the similarity between an item i and its neighbor $j \in S^k(i;u)$ is calculated without considering the content of the neighbor set $S^k(i;u)$.

- Overfitting occurs when a statistical model describes random error or noise instead of the underlying relationship and it can happen by the interpolation weights. For example consider the situation that an item has no useful neighbors that were rated by a particular user. In this scenario it would be best to ignore the neighborhood information and stick to information of the baseline predictors.

- Neighborhood method will not lead to good result if the ratings are substantially different in neighbors ratings.

As a solution to these issues there are several techniques that can be used such as a solution that will solve the problem with uninformative neighbors. The formula below solves some of these limitations [5]:

$$\hat{r}_{ui} = b_{ui} + \frac{\sum\limits_{j \in S^k(i;u)} S_{ij}(r_{uj} - b_{uj})}{\lambda_9 + \sum\limits_{j \in S^k(i;u)} S_{ij}} \tag{2.23}$$

Where λ_9 is used for penalizing the uninformative neighborhood portion such as when $\sum\limits_{j \in S^k(i;u)} S_{ij} \ll \lambda_9$. An accurate estimation of λ_9 will make the recommendation system more accurate and to the point.

Chapter 3

Research Methodology

This chapter discusses the data and methodology that is used in this research. First, an insight to Yelp academic data set is discussed. Yelp Inc. was founded in 2004. This website helps people find local businesses such as doctors, beauty salons, restaurants etc. The website has approximately over 132 *million* visits monthly [11]. In Yelp website every business owner can setup an account to post photos and messages to customers. Yelp's academic data set is used for Yelp challenge [11]. Each year Yelp provides reviews of businesses from real users to academia to explore and research. The main statistics of this data in 2014 is mentioned in table 3.1 :

Table 3.1: General statistics of Yelp data set

Number of Businesses	15,585
Number of Real Users	70,817
Number of Reviews	335,022
Number of Business Attributes	111,561

This data set consists of over $300,000$ customer reviews of businesses throughout several cities in America. These reviews are associated with the reviewer's explicit feelings specified by a rating from 1 (negative) to 5 (positive) stars about the business reviewed. The Yelp data set encompasses several Json files. Here is an example of a user review information:

```
{"votes": {"funny": 5, "useful": 5, "cool": 4 },
"user_id": "3sJ62Mkavx69FBec71agYg", "review_id":
"GXhCVzS14jGRpwusofxFIw", "stars": 5, "date": "2008-08-01",
"text": "Being Indian, and having grown up in London, I have
been pretty spoilt for Indian food.Imagine my dismay when I
moved to Phoenix! I’ve tried many different places here in
town desperately trying to recapture the memory of a good
London curry house but each time I’ve left either disappointed
or clutching my guts fearing what may come. Couple of friends
introduced me to this place a few months ago and I’m stoked.
It’s not as good as the stuff back home. And I’m sure you can
get better in San Francisco or New York. But if you are looking
for something closer to home you can’t go wrong. Plus its got
the real  live Indians eating in there too. Can’t be bad!!",
"type": "review", "business-id": "NCbHGtOP5yJBJsPPaE3X5g" }
```

As seen in the review sample, the user and business names are all encrypted to an ID. The interesting option in these reviews is that each review is subjected to being labeled as useful, funny, or cool by other users. These labels which are mentioned as votes can be considered as an implicit feedback from other users on the reviews.

The "useful" labeled votes can also be comprehended as trust on the review and the expertise of the reviewer on this business and category. One can observe that the aforementioned review has been voted as useful by 5 other users since it is informative and near what most people have experienced with this item, in this case a restaurant.

As an example, say a user named Alice likes reading books and regularly tries different classes in different health centers in town. Now consider we have another user Shane who is a hairdresser herself and has a very good insight about beauty shops in town. There is a higher chance for Alice's reviews on books and fitness centers to receive more votes since she has spend time and know what to look for in these businesses. Also Shane's comments on beauty shops will be more useful because of her experience with different products.

Considering that votes show the expertise of the user on a business, we can have a new recommendation system based on Yelp data set votes. In other words, these votes actually help to find the ratings that are closer to what most people think about these services. Furthermore, this can be very useful since there are the people who read the reviews more than writing them.

There has been studies such as [7] that considers the favorite products of users by the product and service categories they have written reviews for. A more accurate method could be considering the reviews of the user with most "useful" votes in categories of businesses.

Our approach will be introduced in the following section.

3.1 Recommender System Based on Users' Votes

As mentioned earlier each user has the option to vote for a review. These votes can be considered as an implicit feedback as to what the user is good at and actually which criteria is his or her *favorite*. These votes can be used to create different methods for more accurate recommender system.

In this research we have taken into account the “useful” labeled votes, since the usefulness of a review and the expertise of the user's review can be measured by this vote tag. The rating with maximum votes for each item cannot be used as an indicator rate for the item since in this case people with unique tastes are not considered. Moreover, these votes when used as weights for each rating can improve the system performance.

3.2 Method Overview

The first stage of our recommendation system is to consider similar users as the neighbors. This is done by calculating the similarity with the *Pearson Correlation Coefficient* (PCC). The similarity is between the range of -1 to $+1$ ($-1 \leqslant \textit{similarity} \leqslant 1$) where -1 indicates the maximum of negative correlation and $+1$ indicates the maximum positive correlation.

To have the advantage of considering similar users we have defined a threshold value of PCC for the radius of the neighborhood set equal to 0.7(The users with similarity rate equal or more than 0.7 are considered as the neighbors). The second stage of our algorithm is to use the state of the art collaborative filtering method.

Our system is based on the method introduced by Bell and Koren [14]. We have considered user-user based methods. The modification we have made is by considering other users' opinions on the reviews by taking into account the number of votes that each review has as described below:

Pearson Correlation Coefficient is calculated by the formula (3.1).

$$sim(u,v) = \frac{\sum\limits_{s \in S_{uv}} (r_{u,s} - \overline{r}_u)(r_{v,s} - \overline{r}_v)}{\sqrt{\sum\limits_{s \in S_{uv}} (r_{u,s} - \overline{r}_u)^2 \sum\limits_{s \in S_{uv}} (r_{v,s} - \overline{r}_v)^2}} \tag{3.1}$$

Where S_{uv} is the set of all items co-rated by user u and user v. The neighbor set for this study is users with PCC equal or more than 0.7.

Next step is to compute the baseline predictors. The baseline method considers the difference of each user by a coefficient named b_u . To compute the effects associated with items which need to be removed from rating values, parameter b_i is used.

By having the value of b_u and b_i it is easy to reach a value free from user-item interaction that is named b_{ui}. The Bell and Koren method calculates the mean value in the standard method of aggregating the ratings mentioned in (3.2).

$$b_{ui} = \mu + b_u + b_i \tag{3.2}$$

Where μ is :

$$\mu = (1/|S|) \sum_{(c,s) \in S} r_{c,s} \quad , \quad S = \{(c,s) \in S | r_{c,s} \neq \emptyset\} \tag{3.3}$$

One approach to calculate b_i is:

$$b_i = \frac{\sum_{u \in R(i)} (r_{ui} - \mu)}{\lambda_2 + |R(i)|} \tag{3.4}$$

Where $R(i)$ are the users who have rated item i. After computing the value of b_i, we can compute b_u as:

$$b_u = \frac{\sum_{i \in R(u)} (r_{ui} - \mu - b_i)}{\lambda_3 + |R(u)|} \tag{3.5}$$

$R(u)$ in this formula indicates the items rated by user u. The values suggested by Bell and Koren for λ_2 and λ_3 are respectively 25 and 10 for Netflix data [14], and the same values are used for our data set.

Our main contribution in this thesis is to consider opinions of other users on reviews. This is done by computing a weighted μ named μ_w instead of an overall μ. For calculating the μ_w first we need to compute the overall average number of votes. Let $v_{(c,s)}$ indicate the number of votes given for review written by user c for an item s. Then, the mean value of votes is calculated by (3.6):

$$\bar{v} = \frac{\sum\limits_{(c,s) \in S} v_{(c,s)}}{|S|} \quad , \quad S = \{(c,s) \in S | r_{c,s} \neq \emptyset\} \tag{3.6}$$

By having the mean value of votes we can understand which of the ratings are more near to the real experience than others. Popular reviews are those with $v_{(c,s)} \geq \bar{v}$. Therefore, calculating the weighted mean value based on the votes can be done by function below:

$$w_{(c,s)} = \begin{cases} 1 & \text{if } v_{(c,s)} < \bar{v} \\ v_{(c,s)} & \text{if } v_{(c,s)} \geq \bar{v} \end{cases}$$

Where the weights are used for computing μ_w as:

$$\mu_w = \frac{\sum\limits_{(c,s)\in S} r_{(c,s)} w_{(c,s)}}{\sum\limits_{(c,s)\in S} w_{(c,s)}} \quad , \quad S = \{(c,s) \in S | r_{c,s} \neq \emptyset\} \tag{3.7}$$

After computing μ_w , the neighborhood method computation is done by considering PCC as the similarity measure. We have slightly changed the formula in (2.22) to reach the user-user based method as:

$$\hat{r}_{ui} = b_{ui} + \frac{\sum\limits_{v\in S^k(u)} S_{uv}(r_{vi} - b_{vi})}{\sum\limits_{v\in S^k_{(u)}} S_{uv}} \tag{3.8}$$

Since the data is not clustered or categorized the uninformative neighbors have negative impact on the results. To solve this problem the penalizing parameter was used. The value of λ_9 mentioned in (2.23) was considered as 15 by examining a set of numbers between 5 to 30 for which 15 had the best results in terms of error reduction. In general the method can be formalized as (3.9).

$$\hat{r}_{ui} = b_{ui} + \frac{\sum\limits_{v\in S^k(u)} S_{uv}(r_{vi} - b_{vi})}{\lambda_9 + \sum\limits_{v\in S^k_{(u)}} S_{uv}} \tag{3.9}$$

With the penalizing parameter as 15 we have (3.10).

$$\hat{r}_{ui} = b_{ui} + \frac{\sum\limits_{v \in S^k(u)} S_{uv}(r_{vi} - b_{vi})}{15 + \sum\limits_{v \in S^k_{(u)}} S_{uv}} \tag{3.10}$$

Where $\hat{r}_{ui}$ is the predicted value for the unknown rating of item i by user u. $S^k_{(u)}$ denotes the set of similar users to u with the similarity of more or equal to k (in our approach value of k is equal to 0.7). S_{uv} is the similarity measure between user u and user v which is the value of PCC. The flow diagram of the proposed method is shown in figure 3.1.

In the next chapter the case study is presented. Furthermore, analysis and evaluation of the results based on several indicators are discussed.

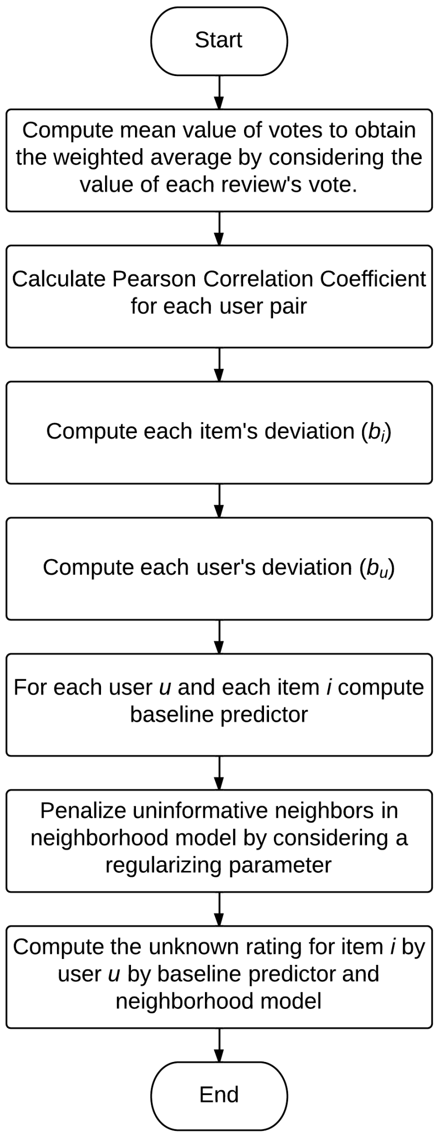
Start
Compute mean value of votes to obtain the weighted average by considering the value of each review's vote.
Calculate Pearson Correlation Coefficient for each user pair
Compute each item's deviation (b_i)
Compute each user's deviation (b_u)
For each user *u* and each item *i* compute baseline predictor
Penalize uninformative neighbors in neighborhood model by considering a regularizing parameter
Compute the unknown rating for item *i* by user *u* by baseline predictor and neighborhood model
End

Figure 3.1: Flow diagram for improved recommender system based on votes and neighborhood model

Chapter 4

Evaluation, Analysis and Comparison

4.1 Evaluation

Yelp academic data set is used to evaluate our approach [11]. We have tested our method in three different scenarios by categorizing the Yelp data set into three cases based on the minimum number of reviews provided by users. In each of these test cases we have randomly consider 70% of the test data set as training and 30% as the evaluating test set which the reviews of the evaluating test set is removed from the system. Then we have applied our method to training data set to predict each of the removed rating r_{ui} of the user u for item i in test set. The predicted rates are estimation of the removed ratings. If the system has a good accuracy, predicted rate for item i by user u should be the same as the removed rating. Error of the system is given by the difference between predicted rating and the real rating as shown in (4.1).

$$e_{ui} = r_{ui} - \hat{r_{ui}} \tag{4.1}$$

Deciding over the accuracy and effectiveness of recommenders only based on (4.1) is not a correct method to assess the accuracy of the system. Therefore, a measurement technique for the accuracy of these predictions is needed which is why we have used a well-known method named *Root Mean Square Error (RMSE)* [9].

RMSE method repeats the procedure of calculating the error for each and every ratings in the test set. The formula used for calculating the RMSE is given by:

$$RMSE = \sqrt{\frac{\sum\limits_{(u,i \in S_T)} (r_{ui} - \hat{r_{ui}})^2}{|S_T|}} \ , \ S_T = \{u, i \in S_T | r_{u,i} \neq \emptyset\} \tag{4.2}$$

Where S_T is the test set. Lower value of RMSE indicates a system with lower error which is a desired characteristic of the recommender system.

In addition to RMSE, we have also used the information-retrieval classification metrics. These metrics evaluate the capacity of the recommender system in suggesting a short list of items to users[8, 26]. These metrics can indicate the probability that the system takes a correct or incorrect decision about the user interest for an item.

Based on the classification methods, the recommendations made can be divided into four kinds. If the user is interested in what the system has suggested to him/her, the system has a true positive (TP), otherwise if the item is uninteresting a false positive (FP) suggestion has been made. If the system cannot predict an interesting item we have a false negative (FN). If the system does not suggest an item not interesting for the user then we have a true negative (TN).

In this research we have considered 3.5 as the threshold value for classifying suggestions as positive or negative as in [6]. The four performance metrics that we have used are Precision, Recall, F measure and Accuracy.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \tag{4.3}$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{4.4}$$

$$F\ measure = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{4.5}$$

$$Accuracy = \frac{TrueNegative + TruePositive}{TotalNumberofUsers} \tag{4.6}$$

Precision measures the proportion of the recommendations that are of interest to the user. *Recall* is the proportion of suggested recommendations which are of interest that appear in top recommendations. *F measure* can be interpreted as a weighted average of the precision and recall. *Accuracy* measures the proportion of correct predictions. All these performance metrics are in range of [0,1] where one corresponds to best performance and zero the worst.

Based on [17, 6], we have categorized our users and items into six categories for our test case with users who provided more than 20 ratings: Heavyrater, Opinionated, Coldstart, Blacksheep users, Controversial items and Niche items which are explained in the following.

Heavyrater users(HR) who provided more than 109 ratings. 109 is the average number of ratings that users provide.

Opinionated users(OP) who provided more than 55 ratings with standard deviation more than 1.2.

Blacksheep users(BS) who provided more than 55 ratings but with mean deviation more than 1.0.

Coldstart Users(CS) who provided less than or equal to 55 ratings.

Controversial items(CI) which received ratings with more than 1.1 standard deviation.

Niche items(NI) which received reviews less or equal to 55 ratings.

Plenty of reviews from Heavyrater users lead to good results in this category. However, it is interesting to see how each system performs in different categories.

4.2 Test Description and Results

In the Yelp data set there are many users who provided no reviews at all or with a very few reviews to be useful to the system. In [7] authors reduced Yelp data set by considering only the users who have at least a minimum of 20 ratings to evaluate their approach. In this study we have considered three different scenarios to study the performance of the system with very few user ratings.

Case 1: Users with more than or equal to 5 reviews.

Case 2: Users with more than or equal to 10 reviews.

Case 3: Users with more than or equal to 20 reviews.

Our reduced data set information on each of these cases are mentioned in table 4.1.

Table 4.1: Different test cases information based on different thresholds for number of users' reviews

Cases	Number of Users	Number of Reviews	Mean Value of Votes
Total Dataset	70,817	335,022	1.2375
Case 1	13,147	230,654	1.4692
Case 2	5,876	183,941	1.6345
Case 3	2,554	139,680	1.8473

Performance of our proposed algorithm mentioned in (3.10) is compared to the Classic Collaborative Filtering method(CCF) as in (2.12), the simple Baseline method (BP) as in (3.2) and the Advanced Neighborhood CF (ANCF) similar to (3.8) without the usage of weighted mean value.

Table 4.2 presents the RMSE of these different methods. Our method is mentioned as *WANCF*, indicating the usage of weighted mean value and the improved advanced neighborhood model.

Table 4.2: RMSE values for different methods

Test Case	CCF	ANCF	BP	WANCF
1	1.148	1.154	1.083	1.082
2	1.147	1.127	1.0493	1.046
3	1.1227	1.0977	1.0201	1.0164

For computing the improvement of method (B) over method (A) for RMSE we have used the formula (4.7).

$$ImprovementPercentage = \frac{(Method(A) - Method(B))}{Method(A)} * 100 \tag{4.7}$$

The improvement of the advanced neighborhood method over classic CF and also the improvement of our method over BP, CCF and ANCF have been computed. In table 4.3 the results of these comparison are presented.

Table 4.3: Improvement percentage of different methods

Test Case	CCF-ANCF	CCF-WANCF	ANCF-WANCF	BP-WANCF
Case 1	-0.5%	5.76%	6.26%	0.13%
Case 2	1.74%	8.84 %	7.24 %	0.30 %
Case 3	2.23%	9.46%	7.40 %	0.35 %

As shown in table 4.3 our system has the maximum of 9.46% improvement over the classical CF and 7.40% from the Bell and Koren's Neighborhood model [14].

We have also compared our method to the baseline predictor to see if we have not over penalized our uninformative neighbors. One of the advantages of our system is that it can give accurate recommendations even for users with few ratings. This is unlike systems such as SMARTERDEALS [7] which needs a minimum of users with more than 20 ratings. SMARTERDEALS method has the $RMSE = 1.07$ as an overall RMSE average which is higher than our result of 1.0164. Our system has an improvement of 5.0 % over SMARTERDEAL average RMSE.

All the figures in the following present the results for Case 3 which is users who have provided more than 20 ratings. Figure 4.1 presents RMSE for all different methods where WANCF has the least value.

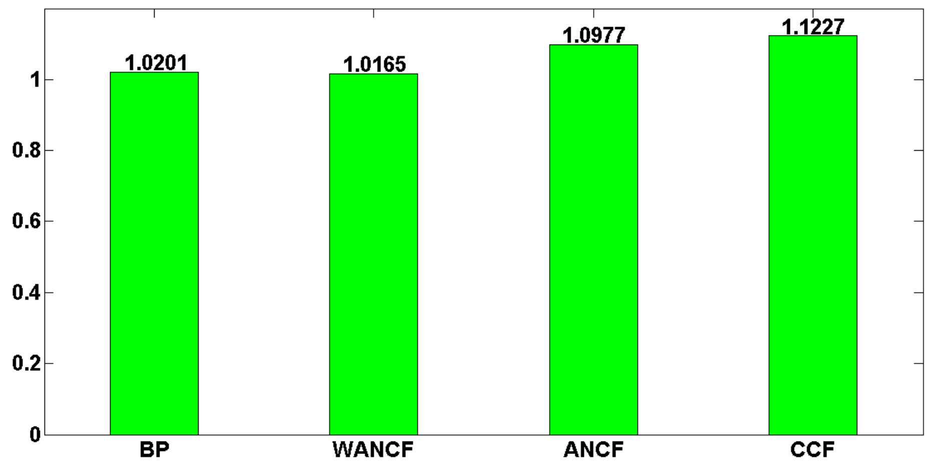

Figure 4.1: RMSE

The next step of our evaluation is to compute the values of Precision, Recall, F measure and Accuracy for these four methods. The results of these performance metrics are shown in figures 4.2 to 4.5.

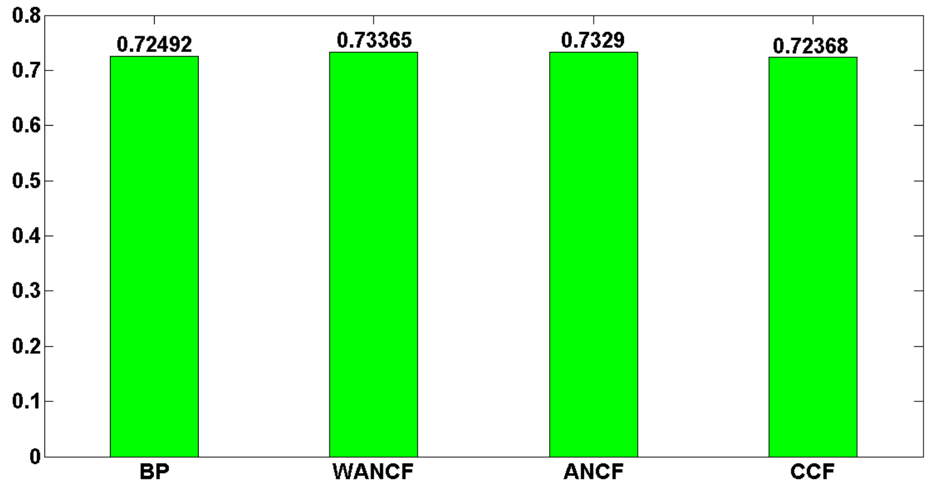

Figure 4.2: Precision

The Precision of WANCF is the highest with 1.2% improvement over BP method. The higher Precision value means that our system has a lower rate of recommending items that are not of any interest to the user.

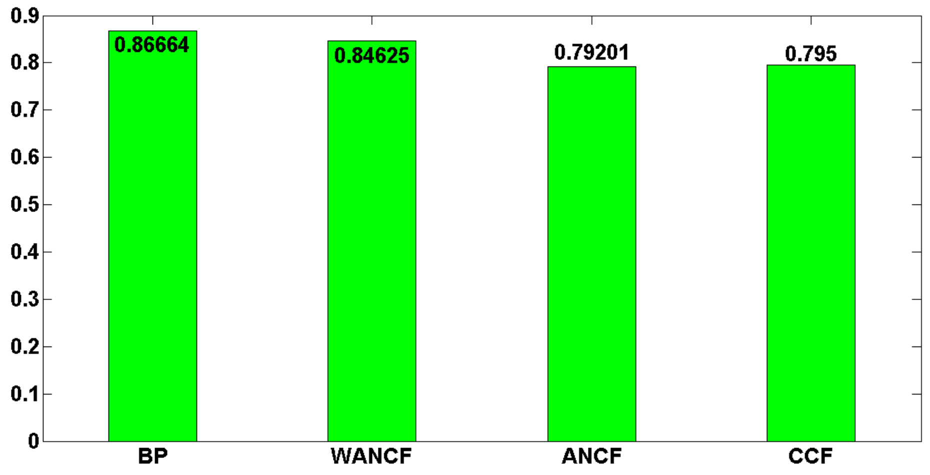

Figure 4.3: Recall

Figure 4.3 illustrates the Recall in which the BP method has the highest Recall value indicating that it has a better top N recommendation list than other methods. However, the Recall measure of our approach is 6.4 % and 6.8 % higher than the ANCF and CCF methods respectively.

F measure is the average of Precision and Recall. BP method with less than 0.5% improvement over WANCF has the highest F measure.

Figure 4.5 demonstrates that WANCF and BP have almost similar Accuracy. Next we have compared these measures with the user and item categories. The result are depicted in figures 4.6 to 4.10.

Figure 4.6 demonstrates the RMSE values for each category. The system is better when it has a smaller value of RMSE. The categories that are worse in terms of RMSE

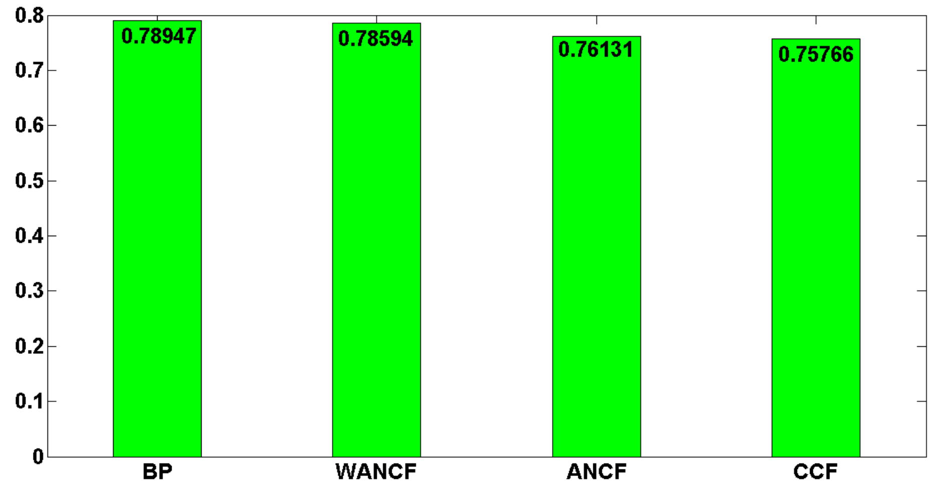

Figure 4.4: F measure

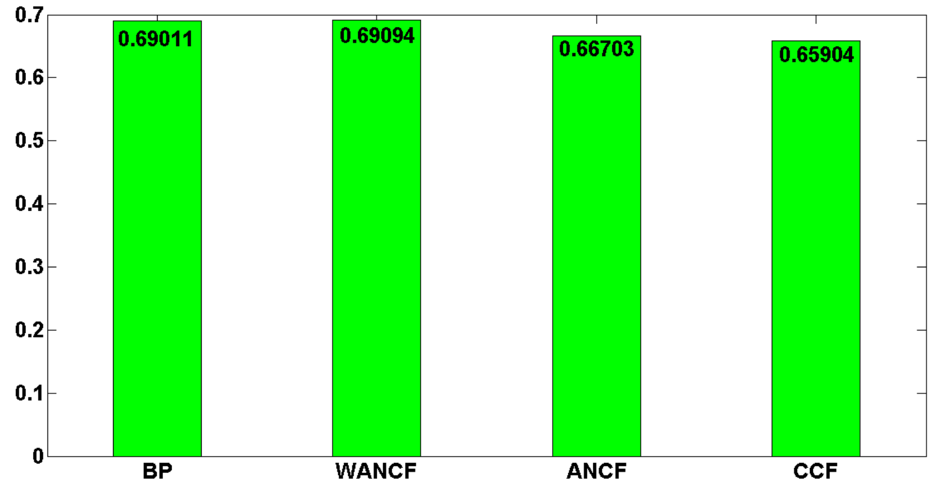

Figure 4.5: Accuracy

are Controversial items, Opinionated users and Blacksheep users. Figure 4.7 shows RMSE improvement over CCF method. Except for the Controversial items, all of the other user categories have improvements more than 3% in WANCF which is quite impressive. The Opinionated user has the highest RMSE improvement equal to 19.3%. Controversial items is the only category in WANCF with no RMSE improvement over

CCF method.

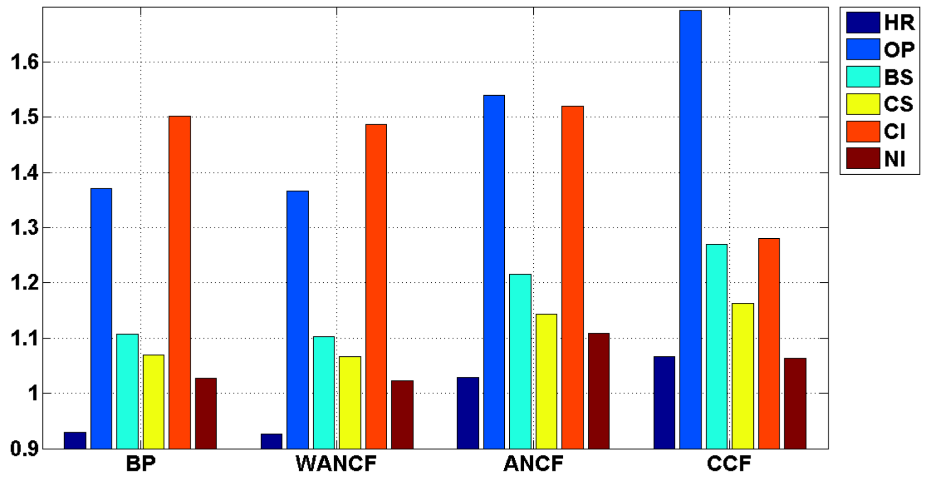

Figure 4.6: RMSE

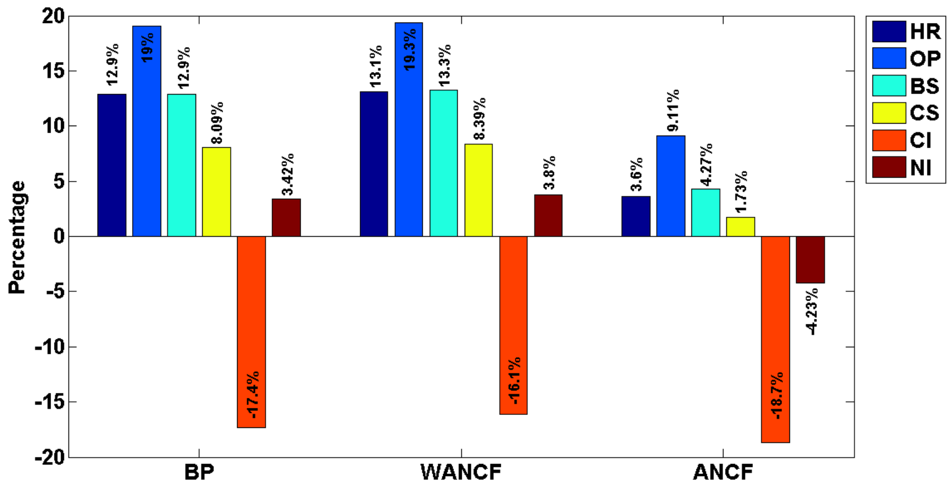

Figure 4.7: RMSE Improvement Over CCF

Figure 4.8 shows the Precision value of each category. Surprisingly, the Precision for Coldstart category is the highest for BP, WANCF and ANCF methods. CS Precision value in ANCF is more than 73%. Moreover, Blacksheep, Heavyrater users and Niche items have higher Precision in all methods except for CCF. In CCF the highest

Precision belongs to Niche items, Controversial items and in third place Coldstart users.

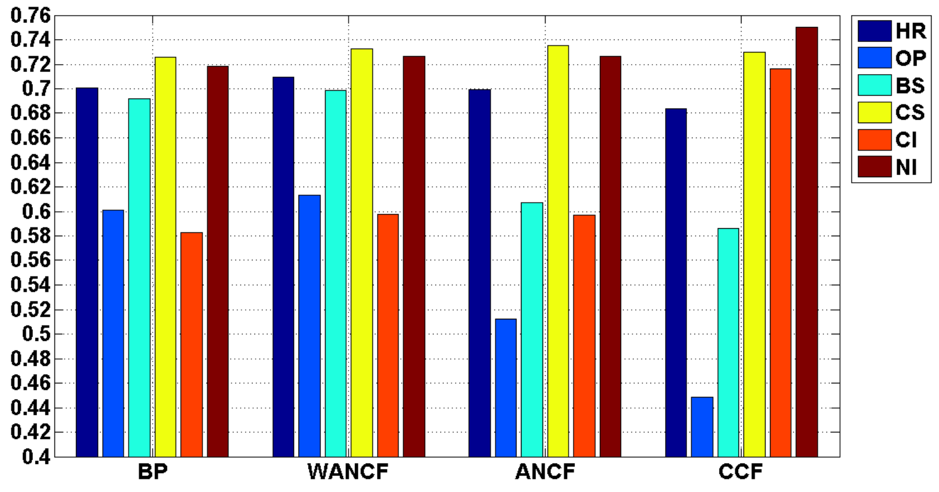

Figure 4.8: Precision

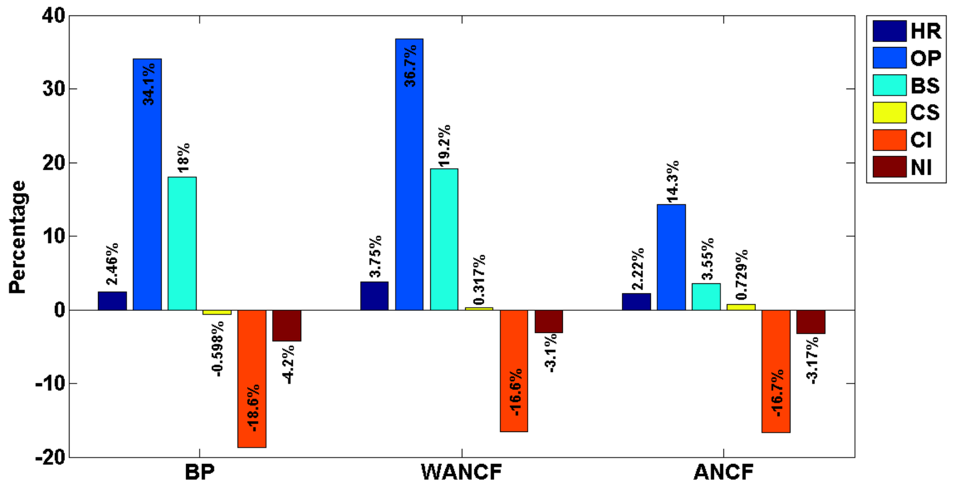

Figure 4.9: Precision Improvement Over CCF

The Precision improvement over CCF method for each category are shown in figure 4.9. Opinionated users is the category with most improvement in all of the three methods over CCF and with the highest improvement rate of 36.7% in WANCF.

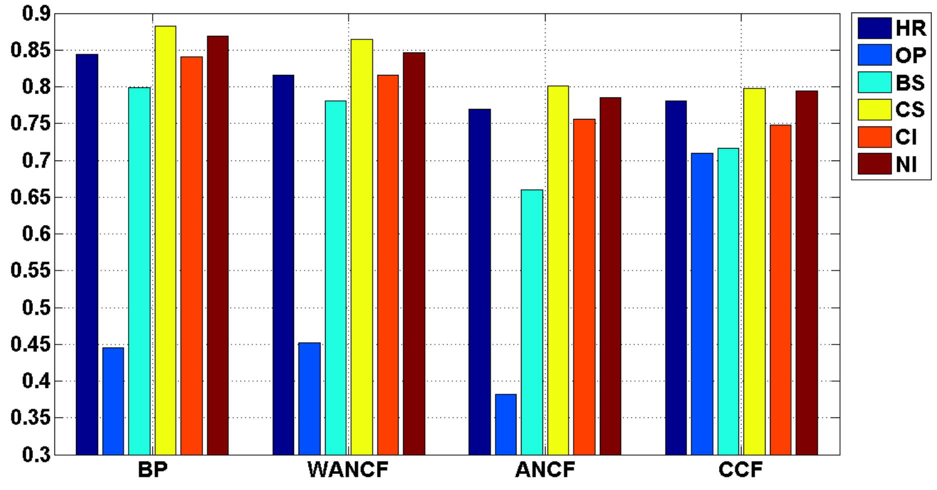

Figure 4.10: Recall

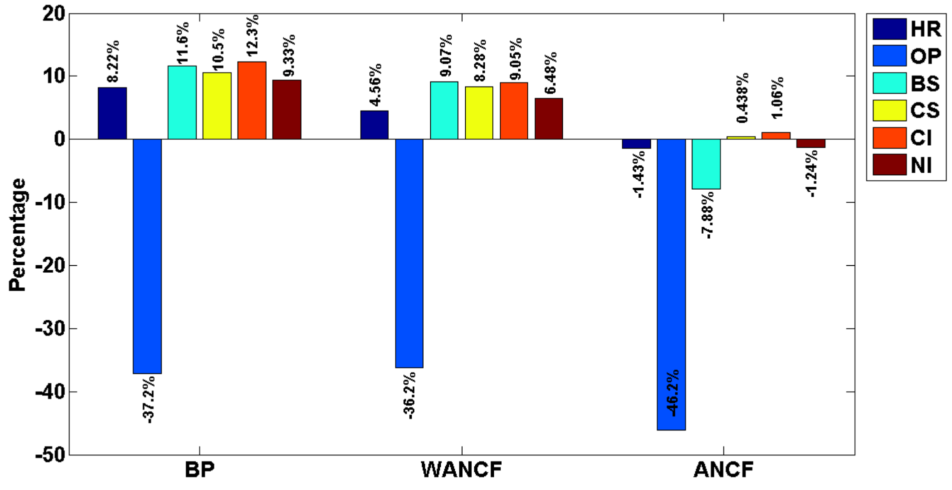

Figure 4.11: Recall Improvement Over CCF

The Recall values for each category are shown in figure 4.10. The Recall for Coldstart users has the highest value for all methods especially for the BP which is near 90%. Based on improvements shown in figure 4.11 for recall values over CCF method, the highest improvement is for Controversial items and the worst is for Opinionated users.

Figure 4.12 demonstrates the values of F measure for each category. Coldstart users still have the highest value, with value of near 80% in BP. The improvement of F measure values for each category over CCF method is shown in figure 4.13. The improvement results are very different for each system. In BP and WANCF, Heavyrater, BlackSheep and Coldstart user categories and Niche items are improved and in ANCF only Coldstart user and Heavyrater categories have an improvement over CCF.

Figure 4.14 presents the values of Accuracy in each category for each method. Surprisingly the highest value of Accuracy is Controversial items recommended by CCF with the value as high as 73%. In the other three methods the highest value is again for Coldstart users. Accuracy improvements over CCF method are shown in figure 4.15. There is a similarity between the Accuracy improvement result and Precision. This is due to the good performance of our system in suggesting true positives and predicting the true negative options. The best improvement is for Blacksheep users in WANCF which is about 13.1%.

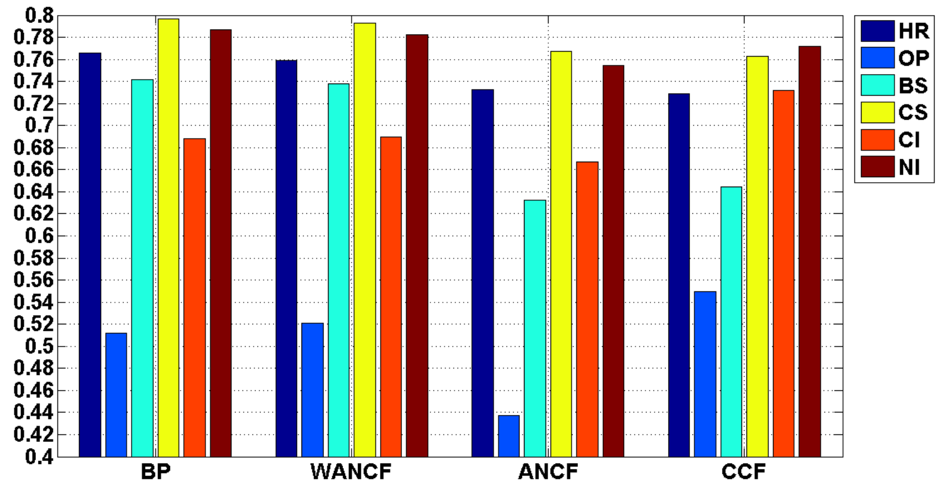

Figure 4.12: F measure

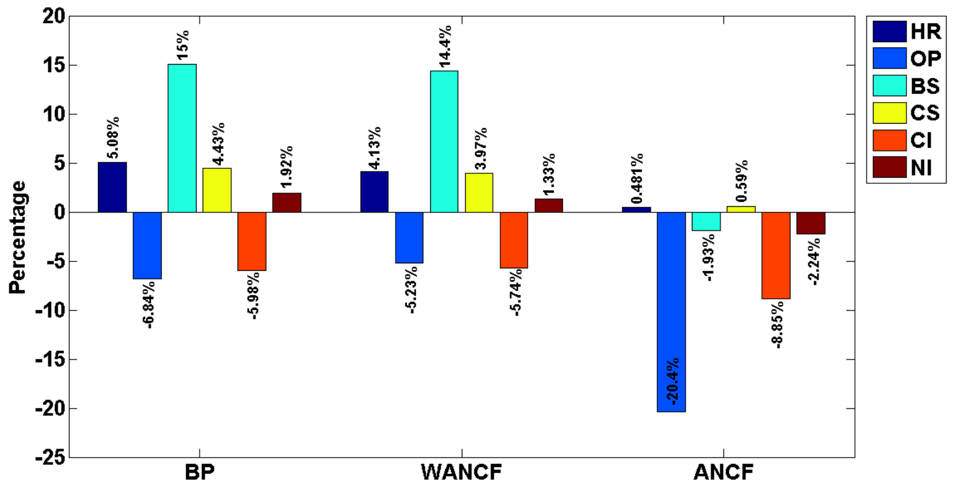

Figure 4.13: F measure Improvement Over CCF

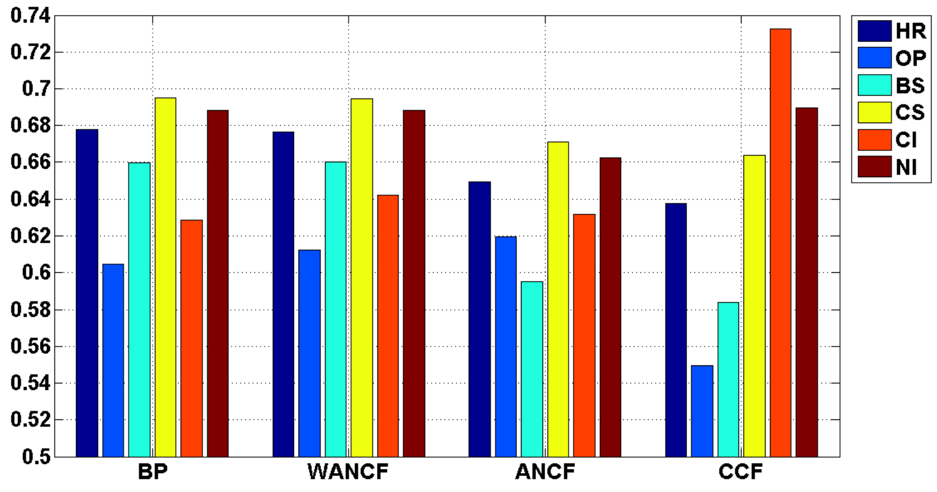

Figure 4.14: Accuracy

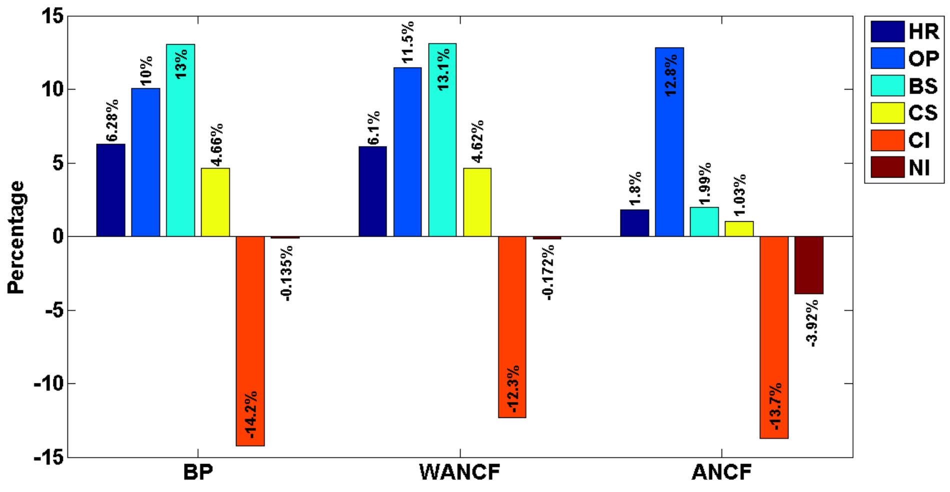

Figure 4.15: Accuracy Improvement Over CCF

Chapter 5

Conclusions

Recommender systems are useful tools for dealing with harnessing the information in the Internet. Its evolution has accompanied with the evolution of the web. Basically every E-commerce business uses different methods to provide the most accurate and useful recommender engines. That is why the accuracy of these systems have a vital role in the evolution of the web. By considering more information of different aspects about each user we will have more accurate recommender engines.

In this thesis we have proposed a method that has used other users' opinions as an implicit feedback for improving the recommender system. We have evaluated our method based on several performance measures including RMSE, Precision, Recall, F measure and Accuracy. The outcome of these evaluations showed great improvement over methods such as Bell and Koren's advanced neighborhood model. The method we have introduced can also be used in systems with users who provided as few ratings as 5, which shows the flexibility of our system. We conclude that WANCF performs better than BP, ANCF and CCF. Our system had promising results specifically in categories such as Blacksheep and Coldstart users which are two of the challenging categories to improve. It is worth mentioning that training of the system is performed offline while the recommendation is online.

Future work involves investigating how clustering of data set can improve recommendations in the categories defined in the thesis. Studying the social network graph presented in the 2014 Yelp data set and its impact on users' reviews as an implicit feedback is another venue for our future work.

Bibliography

[1] Gediminas Adomavicius and YoungOk Kwon. New recommendation techniques for multicriteria rating systems. *Intelligent Systems, IEEE*, 22(3):48–55, 2007.

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

[3] Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. ACM press New York, 1999.

[4] Marko Balabanović and Yoav Shoham. Fab: Content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.

[5] John S Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 43–52. Morgan Kaufmann Publishers Inc., 1998.

[6] Cheng Chen, Lan Zheng, Alex Thomo, Kui Wu, and Venkatesh Srinivasan. Comparing the staples in latent factor models for recommender systems. *SAC, ACM*, 2014.

[7] Sahar Ebrahimi, Norha M Villegas, Hausi A Müller, and Alex Thomo. Smarterdeals: a context-aware deal recommendation system based on the smartercontext engine. In *Proceedings of the 2012 Conference of the Center for Advanced Studies on Collaborative Research*, pages 116–130. IBM Corp., 2012.

[8] Jonathan L Herlocker, Joseph A Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237. ACM, 1999.

[9] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.

[10] Will Hill, Larry Stead, Mark Rosenstein, and George Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 194–201. ACM Press/Addison-Wesley Publishing Co., 1995.

[11] Yelp Inc. Yelp's academic dataset. `http://www.yelp.com/academic-dataset`, 2014.

[12] Joseph A Konstan, Bradley N Miller, David Maltz, Jonathan L Herlocker, Lee R Gordon, and John Riedl. Grouplens: applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–87, 1997.

[13] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434, 2008.

[14] Yehuda Koren and Robert Bell. Advances in collaborative filtering. In *Recommender Systems Handbook*, pages 145–186. Springer, 2011.

[15] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[16] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, 2003.

[17] Paolo Massa and Paolo Avesani. Trust-aware recommender systems. In *Proceedings of the 2007 ACM conference on Recommender systems*, pages 17–24. ACM, 2007.

[18] Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172, 2013.

[19] Julian John McAuley and Jure Leskovec. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *Proceedings of the*

22nd international conference on World Wide Web, pages 897–908. International World Wide Web Conferences Steering Committee, 2013.

[20] Raymond J Mooney, Paul N Bennett, and Loriene Roy. Book recommending using text categorization with extracted information. In *Proc. Recommender Systems Papers from 1998 Workshop, Technical Report WS-98-08*, 1998.

[21] Michael Pazzani and Daniel Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine learning*, 27(3):313–331, 1997.

[22] Michael J Pazzani. A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, 13(5-6):393–408, 1999.

[23] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186. ACM, 1994.

[24] Francesco Ricci, Lior Rokach, and Bracha Shapira. *Introduction to recommender systems handbook*. Springer, 2011.

[25] Gerard Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of*. Addison-Wesley, 1989.

[26] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Application of dimensionality reduction in recommender system-a case study. Technical report, DTIC Document, 2000.

[27] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. pages 285–295. ACM, 2001.

[28] Giovanni Semeraro, Marco Degemmis, Pasquale Lops, and Pierpaolo Basile. Combining learning and word sense disambiguation for intelligent user profiling. In *IJCAI*, volume 7, pages 2856–2861, 2007.

[29] Upendra Shardanand and Pattie Maes. Social information filtering: algorithms for automating word of mouth. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 210–217. ACM Press/Addison-Wesley Publishing Co., 1995.

[30] Amit Sharma and Dan Cosley. Do social explanations work?: studying and modeling the effects of social explanations in recommender systems. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1133–1144. International World Wide Web Conferences Steering Committee, 2013.

[31] David W Vinson and Rick Dale. Valence constrains the information density of messages. *26th APS Annual Convention, May 22-25, San Francisco, California, USA*, 2014.