Incorporating User Reviews as Implicit Feedback for Improving Recommender Systems

Yasamin Heshmat Dehkordi
Department of Computer Science
University of Victoria
Victoria, Canada
yheshmat@uvic.ca

Alex Thomo
Department of Computer Science
University of Victoria
Victoria, Canada
thomo@cs.uvic.ca

Sudhakar Ganti
Department of Computer Science
University of Victoria
Victoria, Canada
sganti@cs.uvic.ca

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 paper we have proposed a new method based on collaborative filtering by considering other users' feedback on each review. To validate our approach we have compared our method with several known methods on Yelp data set. Our algorithm outperforms other approaches in terms of accuracy by as much as 9.5%. We also present our results using comparative analysis for particular categories of users and items. Our algorithm has promising results when handling several difficult user and item categories.**

Keywords—*recommender systems; collaborative filtering; performance metrics; Yelp data set*

I. INTRODUCTION

Since the Netflix Prize competition, Collaborative Filtering (CF) [1] 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. CF 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 [3,4]. As an example there are methods that model how users make their assessment for different purposes, such as suggesting new products they might enjoy [5] or identifying other users who may share similar opinions [6]. 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 the user 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.

Inspired by [7, 8], we define *Coldstart*, *Heavyrater*, *Opinionated*, and *Blacksheep users*, and *Controversial*, and *Niche items*. Except for *Heavyrater users*, the other categories are typically difficult to handle well by recommender systems. We obtain results not only for RMSE, but also for *Precision*, *Recall*, *F measure*, and *Accuracy* for each of the aforementioned categories. Our results are revealing. For instance, for *Blacksheep users*, which are those who go against the mainstream, we surprisingly observed an RMSE improvement of about 13.3% over classical CF. Impressive improvements can also be seen for other categories, such as *Opinionated* and *Coldstart users*. Some more attractive examples are the improvements on *Precision* we observed for *Opinionated users* that are in the order of 36.7% over the classical collaborative filtering. Unlike methods that have a high threshold for the number of reviews of users [9], 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.

More specifically the contributions of this work are as follows.

1. We propose an algorithm based on the user-based collaborative filtering techniques [2] and Koren Bell's algorithm [1, 10] (who are the winners of Netflix prize). 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.

2. We define user and item categories, some of which are difficult to handle by recommender systems (RMSE is high for them), and show that it is for some of these categories that our method really excel and produce promising results.

This paper is organized as follows. Section II presents related background. Section III introduces our proposed algorithm. Section IV presents the analysis and evaluation and case studies. In section V results are compared with well-known methods and finally section VI concludes this research and enumerates avenues of future works.

II. COLLABORATIVE FILTERING

Collaborative recommendation or collaborative filtering (CF) method is based on the quality of items as evaluated by users. These approaches try to predict the utility of items for a particular user based on the items which similar users have chosen in the past. Collaborative filtering techniques based on similarities among users are known as *user-based collaborative filtering* techniques [11,12]. 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.

A. Similarity Computation

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 approaches for calculating similarity are *correlation-based* (cf. Equation (1)) [2, 12, 13] and *cosine-based* (cf. Equation (2)) [8,14].

$$sim(u,v) = \frac{\sum_{i \in C_{uv}} \left(r_{ui} - \bar{r}_u\right)\left(r_{vi} - \bar{r}_v\right)}{\sqrt{\sum_{i \in C_{uv}} (r_{ui} - \bar{r}_u)^2 \sum_{i \in C_{uv}} (r_{vi} - \bar{r}_v)^2}} \quad (1)$$

$$sim(u,v) = \cos(\vec{u},\vec{v}) = \frac{\vec{u}.\vec{v}}{\|\vec{u}\|_2 \times \|\vec{v}\|_2} \quad (2)$$

Equation (1) is known as *Pearson Correlation Coefficient* (PCC). In this equation $sim(u,v)$ is the similarity measure between user *u* and user *v*. r_{ui} and r_{vi} are the ratings given to item *i* by users *u* and *v*, respectively. C_{uv} is the set of items co-rated by both users *u* and *v*. In cosine-based method the items rated by users *u* and *v* are considered as two vectors in an n-dimensional space. Where *n* is the number of all items co-rated by both users *u* and *v*. In both methods for having the set of top similar users to a given user we can define a threshold for the similarity measure (i.e., 0.7 is the threshold considered in this case study).

B. Rating Prediction

Rating prediction is an important phase of the collaborative filtering systems. Rating predictions in user-based method are calculated by considering previously rated items by the users. The value of the unknown rating r_{ui} for user *u* on item *i* can be calculated as an aggregate of the ratings from the most similar users for the targeted item *i* [2]. The easiest method for aggregating is as: $r_{ui} = aggr\ r_{v \in \hat{C}_{vi}}$, where $\hat{C}$ is a group of *N* users that are most similar to user *u* who have rated item *i*. The simplest method for considering all the ratings of the similar users is to have an average of their ratings: $r_{ui} = \frac{1}{N \sum_{v \in \hat{C}} r_{vi}}$. However, the most popular method is to have a weighted sum with similarities being the weights in the formula: $r_{ui} = k \sum_{v \in \hat{C}} sim(u,v) \times r_{vi}$, where k is a normalizing factor that is $k = \frac{1}{\sum_{v \in \hat{C}} |sim(u,v)|}$, and $sim(u,v)$ is the similarity of user *u* to *v*. Each user has his/her own standards for ratings, which the aggregation methods mentioned above do not consider. To fix the problem, deviations from the average ratings of users are considered by using adjusted weighted sum (cf. Equation (3)) [12].

$$r_{ui} = \bar{r}_u + k \sum_{v \in \hat{C}} sim(u,v) \times (r_{vi} - \bar{r}_v) \quad (3)$$

C. Classical Collaborative Filtering (CCF)

In this paper we have named the traditional user-based collaborative filtering method in [11, 12] as CCF. This method uses the similarities among users to recommend items. The similarity level between a pair of users is calculated based on similar ratings and choices as mentioned in Equation (1). Equation (3) is the formula used for rating prediction in CCF.

D. Baseline Predictors

CCF model and all of the traditional CF methods capture the interactions between users and items that produce the different rating values. In [1,10] Koren and Bell argue that much of the observed rating values are due to effects associated with either users or items, independently of their interaction, such as tendencies of some users to give higher ratings than other or in case of items some items receive higher ratings than others. For considering these effects they have introduced *baseline predictors.* Baseline predictor for an unknown rating r_{ui} is denoted by b_{ui} which is the parameter that encapsulates the effects that do not involve user-item interaction. Equation (4) shows the calculation of b_{ui}.

$$b_{ui} = \mu + b_u + b_i \quad (4)$$

μ is the average rating over all items rated by all of the users, b_u and b_i indicate the observed deviation of user *u* and item *i*, from the average respectively:

$$b_i = \frac{\sum_{u \in R(i)} (r_{ui} - \mu)}{\lambda_2 + |R(i)|} \quad (5)$$

$$b_u = \frac{\sum_{i \in R(u)} (r_{ui} - \mu - b_i)}{\lambda_3 + |R(u)|} \quad (6)$$

Calculation of b_i and b_u are provided in details in [1]. Koren and Bell demonstrated that 25 and 10 are typical values for λ_2 and λ_3, respectively. We used the same values for the Yelp data set.

E. Advanced Neighborhood Collaborative Filtering

To reach unknown ratings predicted based on using similarity measures among users modified Equation 5.17 of [15] was used to compute unknown ratings as in Equation (7).

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{v \in S^k(u)} s_{uv} (r_{vi} - b_{vi})}{\sum_{v \in S^k(u)} s_{uv}} \quad (7)$$

v indicates users with a similarity with user *u* higher than the threshold *k* (in our case 0.7), and $S^k(u)$ denotes the set of all users similar to *u*. We call this approach Advanced Neighborhood Collaborative Filtering (ANCF).

III. Weighted Advanced Neighborhood CF (WANCF)

Our proposed algorithm is an improvement of ANCF by considering other users' opinions as implicit feedback for improving the recommender system in terms of accuracy and other performance metrics. 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_{ui} indicate the number of votes given for review written by user *u* for item *i*. Then, the mean value of votes is calculated by Equation (8).

$$\bar{v} = \frac{\sum_{(u,i)} v_{ui}}{|C|} \quad , C = \{(u,i) \in C | r_{ui} \neq \emptyset\} \tag{8}$$

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 $v_{ui} \geq \bar{v}$. Therefore, calculating the weighted mean value based on the votes can be done by the introducing weights as function below:

$$w_{ui} = \begin{cases} 1 & if \ v_{ui} < \bar{v} \\ v_{ui} & if \ v_{ui} \geq \bar{v} \end{cases} \tag{9}$$

Where the weights are used for computing μ_w as in Equation (10):

$$\mu_w = \frac{\sum_{(u,i) \in C} r_{ui} w_{ui}}{\sum_{(u,i) \in C} w_{ui}} \tag{10}$$

After computing μ_w we have computed $\acute{b}_{ui}$ as:

$$\acute{b}_{ui} = \mu_w + b_u + b_i \tag{11}$$

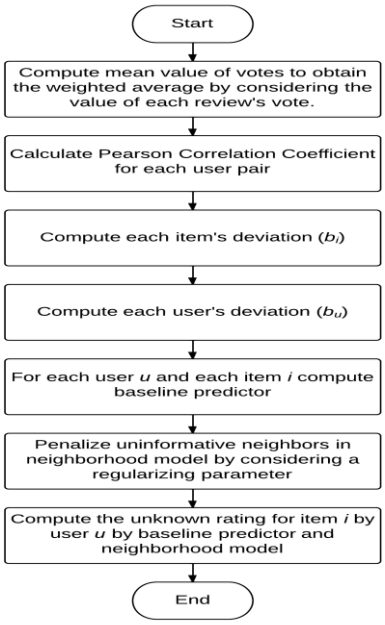

Fig. 1. Flow diagram of WANCF method

The rating prediction formula is similar to equation (7) with $\acute{b}_{ui}$ instead of b_{ui}.We also have used penalizing parameter (λ_9) introduced in Equation 5.18 of [1] for penalizing the neighborhood portion when there is not much neighborhood information which in these situations $\sum_{v \in S^k(u)} S_{uv} \ll \lambda_9$. The value of λ_9 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 Equation (12).

$$\hat{r}_{ui} = \acute{b}_{ui} + \frac{\sum_{v \in S^k(u)} S_{uv} (r_{vi} - \acute{b}_{vi})}{15 + \sum_{v \in S^k(u)} S_{uv}} \tag{12}$$

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* in Equation (1). Flow diagram for improved recommender system based on votes and neighborhood model is presented in Fig. 1.

IV. Analysis & Evaluation

A. Performance Metrics

Yelp academic data set is used to evaluate our approach [16]. 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 $e_{ui} = r_{ui} - \hat{r}_{ui}$. Deciding over the accuracy and effectiveness of recommenders only based on one error is not a correct method to assess the accuracy of the system. Therefore, we have used a well-known method named *Root Mean Square Error* (RMSE) [17]. 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:

$$RMSE = \sqrt{\frac{\sum_{(u,i \in S_T)} (r_{ui} - \hat{r}_{ui})^2}{|S_T|}} \tag{13}$$

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 [18, 19]. 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 [7]. The four performance metrics that we have used are *Precision*, *Recall*, *F measure* and *Accuracy*.

$$Precision = \frac{TruePositive}{TruePositive+FalsePositive} \quad (14)$$

$$Recall = \frac{TruePositive}{TruePositive+FalseNegative} \quad (15)$$

$$F\ measure = 2 * \frac{Precision*Recall}{Precision+Recall} \quad (16)$$

$$Accuracy = \frac{TrueNegative+TruePositive}{TotalNumberofUsers} \quad (17)$$

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.

B. Test Cases

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 [9] 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.

TABLE I. TEST CASES INFORMATION

Cases	No. of Users	No. of Reviews	Mean Value of Votes
Total Data set	70,817	335,022	1.2375
1	13,147	230,654	1.4692
2	5,876	183,941	1.6345
3	2,554	139,680	1.8473

- 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 I.

C. User and Item Categories

Based on [7, 8], 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.

V. RESULTS & DISCUSSION

Performance of our proposed algorithm mentioned in Equation (12) is compared to the Classic Collaborative Filtering method (CCF) as in Equation (3), the simple Baseline method (BP) as in Equation (4) and the Advanced Neighborhood CF (ANCF) mentioned in Equation (7). Table II 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 II. 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 RMSE improvement percentage of method (B) over method (A) formula (18) has been used.

$$Improvement\ Percentage = \frac{Method(A)-Method(B)}{Method(A)} \times 100 \quad (18)$$

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 III the results of these comparison are presented.

TABLE III. IMPROVEMENT PERCENTAGE OF DIFFERENT METHODS

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

As shown in Table III our system has the maximum of 9.46% improvement over the classical CF and 7.40% from the Bell and Koren's Neighborhood model (ANCF) [1]. 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 [9] 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.

In Table IV the values of *Precision*, *Recall*, *F measure* and *Accuracy* for all four methods are computed. Note that these are the results for Case 3 which is users who have provided more than 20 ratings.

TABLE IV. PERFORMANCE METRICS OF DIFFERENT METHODS

Method	Precision	Recall	F measure	Accuracy
CCF	0.72368	0.795	0.75766	0.65904
ANCF	0.7329	0.79201	0.76131	0.66703
BP	0.72492	0.86664	0.78947	0.69011
WANCF	0.73365	0.84625	0.78594	0.69094

TABLE V. PERFORMANCE METRICS IMPROVEMENT PERCENTAGE OF DIFFERENT METHODS

Methods Comparison	Precision	Recall	F measure	Accuracy
ANCF- CCF	1.27%	-0.38%	0.48%	1.21%
BP-CCF	0.17%	9.01%	4.2%	4.71%
WANCF-CCF	1.38%	6.45%	3.73%	4.84%
BP-ANCF	-1.1%	9.42%	3.7%	3.46%
WANCF-ANCF	0.11%	6.84%	3.24%	3.58%
WANCF-BP	1.20%	-2.35%	-0.45%	0.12%

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. 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 6.45 % and 6.84 % higher than the CCF and ANCF 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*. WANCF and BP have almost similar *Accuracy* with WANCF being 0.12% higher.

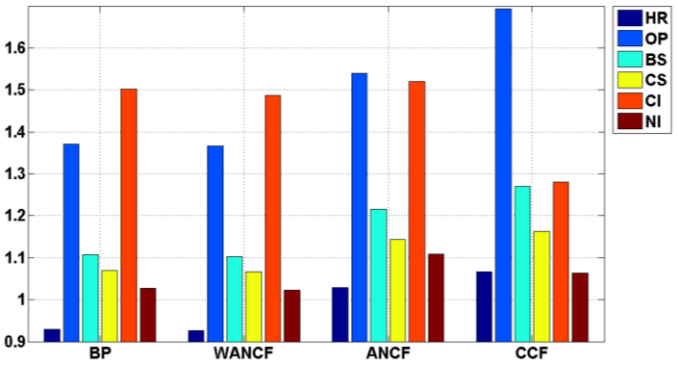

Fig. 2. RMSE values of each category for the four different methods

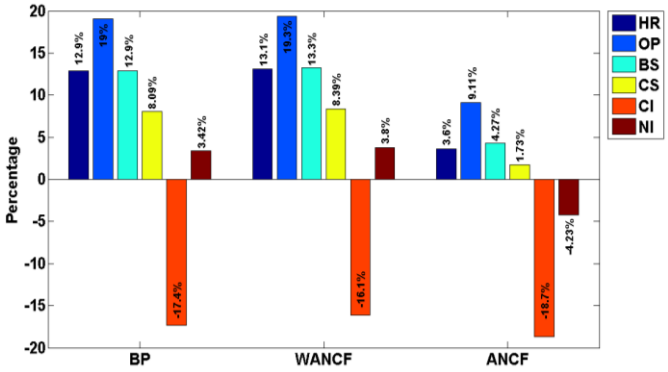

Fig. 3. RMSE Improvement over CCF

Next we have compared the performance metrics with the user and item categories. The results are depicted in Figs. 2 to 11. Fig. 2 demonstrates the RMSE values for each category. The categories that are worse in terms of RMSE are *Controversial items*, *Opinionated users* and *Blacksheep users*. Fig. 3 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 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*. The *Precision* improvement over CCF method for each category are shown in Fig. 5. *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.

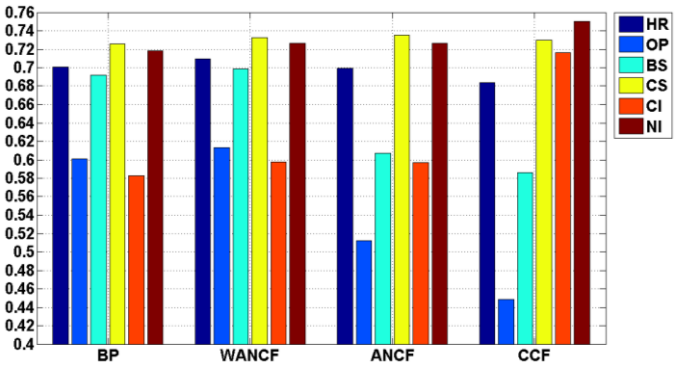

Fig. 4. Precision values of each category for the four different methods

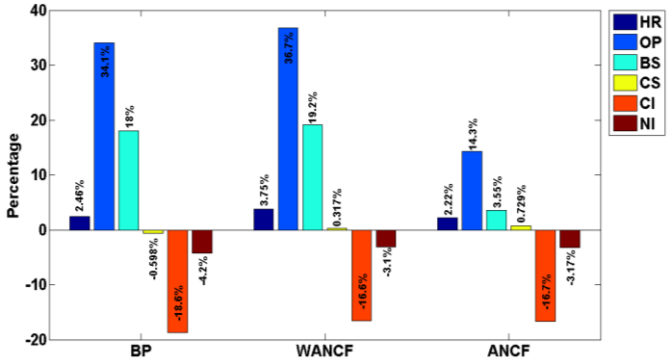

Fig. 5. Precision Improvement over CCF

The *Recall* values for each category are shown in Fig. 6. 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 Fig. 7 for *Recall* values over CCF method, the highest improvement is for *Controversial items* and the worst is for *Opinionated users*. The *BS* and *CS* category in our method shows an improvement of 9.07% and 8.28% over CCF which is impressive for these two challenging user categories.

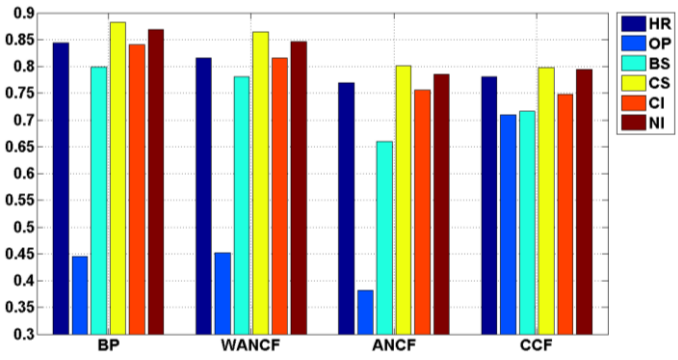

Fig. 6. Recall values of each category for the four different methods

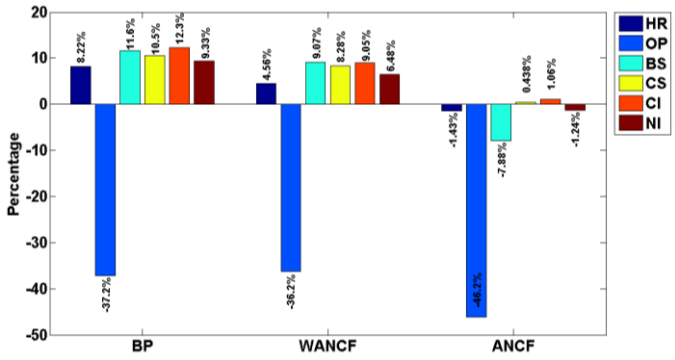

Fig. 7. Recall Improvement over CCF

Fig. 8 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 Fig. 9. 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. The *Blacksheep user* category has revealing improvement in both BP and WANCF method equal to 15% and 14.4% over CCF respectively.

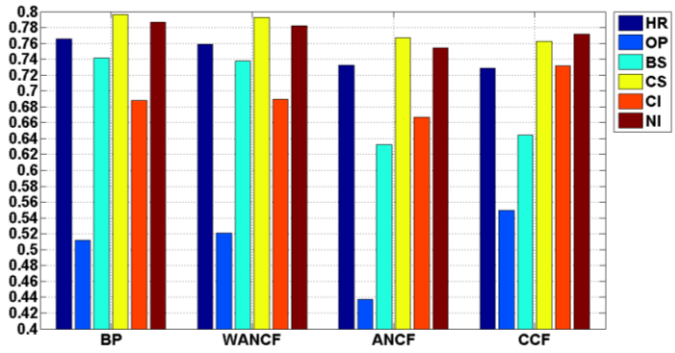

Fig. 8. F measure values of each category for the four different methods

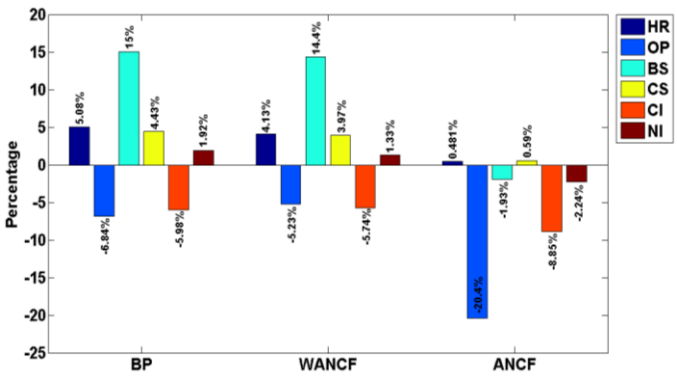

Fig. 9. F measure Improvement over CCF

Fig. 10 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 Fig. 11. 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%.

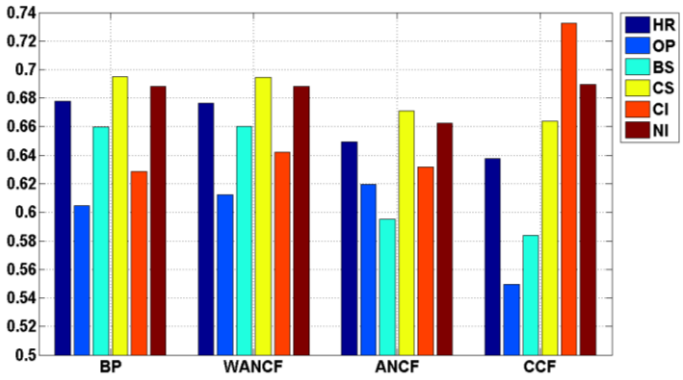

Fig. 10. Accuracy values of each category for the four different methods

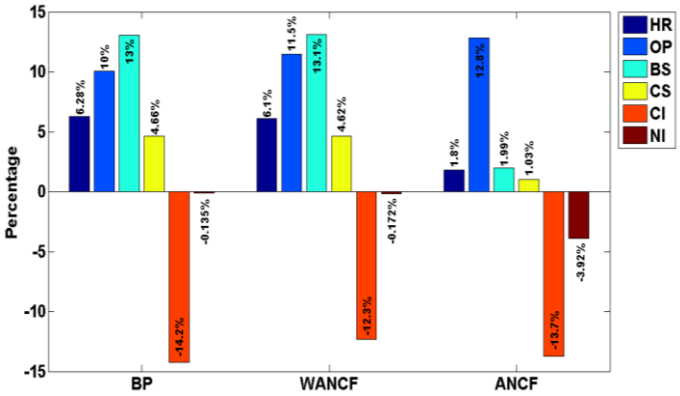

Fig. 11. Accuracy Improvement over CCF

VI. Conclusion & Future Works

In this paper 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 most 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 paper. 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.

References

[1] Yehuda Koren and Robert Bell. Advances in collaborative filtering. In Recommender Systems Handbook, pp. 145-86. Springer, 2011.

[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, vol.* 17, *no.* 6, *pp.* 734-749, 2005.

[3] 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, pp. 165-172, 2013.

[4] David W Vinson and Rick Dale. Valence constrains the information density of messages. *In Proceedings of the 36th Annual Conference of the Cognitive Science Society*. Austin, TX: Cognitive Science Society, 2014.

[5] 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*, pp. 897-908. International World Wide Web Conferences Steering Committee, 2013.

[6] Amit Sharma and Dan Cosley. Do social explanations work?: studying and modeling the e ects 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.

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

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

[9] 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*, pp. 116-130. IBM Corp., 2012.

[10] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer, vol.* 42, *no.* 8, pp. 30-37, 2009.

[11] 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.

[12] 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*, pp. 175-186. ACM, 1994.

[13] 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.

[14] 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, pp. 43-52. Morgan Kaufmann Publishers Inc., 1998.

[15] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE, vol.* 7, *no.* 1, pp. 76-80, 2003.

[16] Yelp Inc. Yelp's academic dataset [oline] 2014, http://www.yelp.com/academic-dataset, (Accessed: 25 May 2014).

[17] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. Evaluating collaborative filtering recommender systems. ACM *Transactions on Information Systems* (TOIS), *vol.* 22, *no.* 1, pp. 5-53, 2004.

[18] 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*, pp. 230-237. ACM, 1999.

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