Recommender systems based on opinion mining and deep neural networks

Jia Li¹, and YongJian Yang^{2,*}

¹College of Software, Jilin University, Changchun, China

²College of Computer Science and Technology, Jilin University, Changchun, China

Abstract. To address rating sparsity problem, various review-based recommender systems have been developed in recent years. Most of them extract topics, opinions, and emotional polarity from the reviews by using the techniques of text analysis and opinion mining. According to existing researches, review-based recommendation methods utilize review elements in rating prediction model, but underuse the actual ratings provided by users. In this paper, we adopt one lexicon-based opinion mining method to extract opinions hidden in reviews, and also, we combine opinions with actual ratings. In addition, we embed deep neural networks model which breaks through the limitation of traditional collaborative filtering. The experimental results based on two public datasets indicate that this personalized model provides an effective recommendation performance.

1 Introduction

Currently, most of the recommendation methods utilize actual ratings generated by users to infer user preferences. Among all of those techniques, collaborative filtering (CF) receives extensive attention and progress, meanwhile, matrix factorization (MF) becomes the first choice for CF [1], which is widely used and performs well. MF factorizes the rating matrix into two matrices containing latent features, and then, MF infers the unobserved ratings based on inner product of two matrices. However, like any technology there are limits: the performance is weak when the data is sparse, which also existed in most recommendation techniques. Moreover, inner product function limit the expressiveness of MF [2]. This paper devotes to address the above issues and improve recommendation effect.

In reality, data is sparse and full of noise, we need to focus on how to utilize low-quality data to improve the quality of recommendation. Thus, a variety of review-based recommender systems have been proposed in recent years, researchers intended to utilize the effective information extracted from reviews for addressing rating sparsity problem. To this end, they used text analysis and opinion mining methods to extract topics, opinions, sentiments, contextual information from textual reviews. Moreover, it has been proved that these review elements enable to alleviate rating sparsity and cold-start problems [3]. According to existing review-based recommendation researches, we detect that most of them exploit review elements for rating prediction models, but overlook the other effective information in dataset. Such as [4-5], they find the importance of users reviews, however, they ignore ratings provided by users.

On the other hand, MF uses inner product to infer complex user-item interactions in low-dimensional latent space, which may lead to limitation. In this paper, we address this issue by utilizing a neural network model, which replaces inner product and learns the user-item interaction function [2].

The main contributions of this paper as follows:

- Propose a general, expandable and personalized rating prediction framework, which consists of opinion mining technique and deep neural networks.
- Make full use of existing data by combining review elements with actual ratings together, experimental results indicate the improvement is effective.
- Multiple experiments on two datasets to demonstrate the effectiveness of the proposed approach.

This paper is organized as follows. We review the related work in Section 2 and present our proposed framework in Section 3, we conduct experiments and draw conclusions in Section 4. In Section 5, we conclude this paper.

2 Related works

In recent years, many researchers have tried to improve the predictive effect by utilizing the effective information in reviews. [5] points out that opinions extracted from reviews express more emotions than actual ratings, they utilize review elements in rating prediction model but ignore users' ratings information. [4] uses a collaborative filtering algorithm based on nearest neighbor, and takes the opinions of reviews as input data. As we know, they also do not consider actual ratings provided by users. Certainly, there are some

* Corresponding author:yyj@jlu.edu.cn

researches such as [6-7] utilize both of those two types of data. [8] claims that blended data combining virtual scores (numerical value of opinions extracted from reviews) with actual ratings may lead to better recommend results. [7] makes full use of the votes information, each review receives a certain number of votes, which denote whether a reader found the review helpful or not. This framework allocates different weights to the users' ratings which depends on the count of votes. [6] assigns weights to the users' ratings as well, but utilizes the aspects discussed in reviews. [9] calculates the overall opinion of each review, and learns prediction model based on opinions and ratings separately. However, they make use of traditional matrix factorization to learn the interaction function.

This paper devotes to alleviate rating sparsity, to this end, we adopt opinion mining to extract opinions and sentiments from reviews, and also combine these opinions information with actual ratings. Meanwhile, we want to offer a better recommender performance by addressing the limitation of MF, for this purpose, we use a collaborative filtering technique which modeled by neural networks.

3 System models

In this section we present a framework containing two parts: opinion mining and deep neural networks. We adopt one lexicon-based method to analyze key factors of reviews, and we make use of Neural Collaborative Filtering (NCF) [2] to learn rating prediction model.

3.1 Opinion mining

Opinion mining or sentiment analysis could infer latent emotions, feelings, views and attitudes based on words expressed by users [10]. In this work we identify the opinion words or phrases that occur in reviews and sentiment lexicon, and then, we calculate the total opinion score or sentiment score of each review.

3.1.1 Sentiment lexicon

The sentiment lexicon we use is obtained from Tidytext, which is the toolkit of R. This toolkit contains three sentiment lexicons for sentiment analysis, they are NRC, AFINN and BING. We adopt AFINN [11] lexicon finally, which consists of 2500 words and sentiment scores.

	word	sentiment	lexicon	score
	<chr>	<chr>	<chr>	<int>
1	abandon	<NA>	AFINN	-2
2	abandoned	<NA>	AFINN	-2
3	abandons	<NA>	AFINN	-2
4	abducted	<NA>	AFINN	-2
5	abduction	<NA>	AFINN	-2
6	abductions	<NA>	AFINN	-2
7	abhor	<NA>	AFINN	-3
8	abhorred	<NA>	AFINN	-3
9	abhorrent	<NA>	AFINN	-3
10	abhors	<NA>	AFINN	-3

Fig. 1. Structure of AFINN lexicon.

As shown in Fig.1., AFINN has no sentiment category (all are NA). The last column is the sentiment

score, each word corresponds to one score, the range of values between -5 and 5.

3.1.2 Extract opinions from reviews

Each review is defined as R_{ui} , which consists of a list of words ($w_1, w_2, \dots, w_n$), u represents user and i denotes item. For each word in reviews, we define its semantic orientation as $O(w)$, as shown in equation (1). $O(w)$ equals to sentiment score $S(w)$ multiply by one coefficient C_w . the value of coefficient C_w is 1 or -1, which represents whether there is a negative word before the opinion word.

$$O(w) = C_w S(w) \quad (1)$$

In this paper, we adopt one simple opinion mining method, the procedures adopted include the following steps:

- Token review by using punctuation marks, and also filter out the stop words and non-character words.
- Identify words which occur both in review and sentiment lexicon.
- Summarize the semantic orientations of all opinion words and calculate its mean value, then we obtain the overall opinion score of each review.

The overall opinion score of each review $S(R_{ui})$ can be expressed as

$$S(R_{ui}) = \frac{\sum O(w)}{N} \quad (2)$$

In equation (2), N is the number of opinion words. All in all, we get the blended data after combining opinion scores with users' ratings. There is a weight coefficient that controls the proportion of two types of data, the range of this coefficient between 0 and 1. For the convenience of observing the experimental results, we also generate single opinion scores data and users' ratings data.

3.2 Deep neural networks

Recently, neural networks have been found that playing a pivotal role in several domains, including computer vision, speech recognition and text processing [12, 13]. However, in most cases, neural networks play a supporting role in recommender systems.

We embed the Neural Collaborative Filtering (NCF) [2] framework in our approach, NCF replaces inner product and models the user-item latent structures by neural networks. This framework models the collaborative filtering effect and breaks through the limitation of traditional matrix factorization method. NCF combines two techniques: GMF (Generalized Matrix Factorization) and MLP (Multi-Layer Perceptron). GMF models the interactions by using linear kernel, MLP uses non-linear kernel to learn the interaction function. NCF framework allows GMF and MLP trained separately, and then inputs the training results into the last hidden layer.

4 Experiments

4.1 Datasets

We experiment with two public datasets: Yelp and Amazon, Table 1 summarizes the characteristics of these two datasets.

1. Yelp. The dataset of Yelp Challenge, which has been widely used in recommender systems. We use one sample dataset, including 280K reviews, over 140K users and 11K items. The main format of the dataset is user-item-rating-review-time.

2. Amazon. This is the Amazon electronic dataset, including users' ratings and reviews. This dataset is larger and sparser than Yelp, there are over 190K users and 63K items.

We filter out the users having fewer than 10 reviews, and the results are presented in the Table 1.

Table 1. Statistics of datasets in evaluation.

Dataset	Interaction	Item	User	Sparsity
Yelp	62,724	8,301	3,018	99.74%
Amazon	181,915	40,662	3,621	99.87%

As NCF model addresses the recommendation with implicit feedback, we need to transform each interaction into implicit data, but at the same time we have to keep the differences among the different types of data. To this end, we sort the value of each interaction and filter out the low-grade data according to the certain proportion. In the end, the remaining data is the final experiment data. For each user, we take his latest interaction as the test data, hence, the remaining data belong to training set. Meanwhile, we collect the negative instances from the item list that not interacted by the user.

4.2 Evaluation metrics

We adopt the leave-one-out evaluation standard [14] to measure the recommendation effect of our framework. For the sake of time cost, we extract only 100 samples from items that have not rated by users, and this collection is the negative instances set [15]. Therefore, the prediction task is to rank the test item among those 100 items.

We utilize Hit Ratio (HR) and Cumulative Gain (NDCG) [16] to judge the performance of a ranked list. HR measures whether the test item is in the list of Top-N, and NDCG measures the quality of rank. NDCG indicates the position of the hit item in the list by assigning different scores to the test item in top recommendation list. The higher position the item in, the higher weight score will be allocated.

4.3 Baselines

We compare our proposed approach with the following commonly used methods:

1. Itempop[17]. This method recommends items based on popularity. Itempop is not a personalized

recommend technique, but it's a kind of widely used and basic method in recommender systems.

2. ItemKNN [18]. ItemKNN is a standard collaborative filtering method. We use cosine similarity to measure the similarity between items.

3. PureSVD [17]. This is an advanced algorithm in top-N recommendation, and it uses the singular value decomposition on the whole matrix.

4.4 Parameter settings

As we mentioned above, we define a weight coefficient to control the proportion of opinion scores and actual ratings in blended data, the default value of this coefficient to 0.3.

There are many parameters in NCF model such as layers number of MLP, learning rate, batch size, etc. We set the default value of batch size to 256, the default layers of MLP is {64, 32, 16, 8}, and there are four negative instances per positive instance, the default value of learning rate is 0.001 and prediction factors is 8. We allow GMF and MLP to contribute equally. As the experiment progresses, we consciously adjust these parameters to achieve the optimal effect.

4.5 Results

4.5.1 Opinion result

After extracting opinions from reviews, we normalize two result sets and obtain the distribution of the opinion scores data and actual ratings data. Let's take Yelp dataset for instance, the mean of actual ratings data is 0.66 and the standard deviation is 0.28; The mean of opinion scores data is 0.603, and the standard deviation is 0.135.

We can infer that the proportion of high polarity sentiments is small in opinion scores data, most of scores concentrate near 0.603, actual ratings data have a larger standard deviation. It's obvious that the opinions extracted from reviews is more neutral than users' ratings. We compare the results among three types of input data, they are single ratings data, single opinion scores data and blended data, all parameters remains unchanged during the experiment. The summarized information are showed in the Table 2. Without special mention, blended data is the default input data.

Table 2. Results of three types of data.

Data type	Yelp		Amazon	
	HR	NDCG	HR	NDCG
Ratings data	0.5460	0.3423	0.4348	0.2564
Opinion scores	0.5435	0.3365	0.4350	0.2572
blended data	0.5697	0.3565	0.4352	0.2578

As can be seen in Table 2, we obtain the best recommender performance based on the blended data, but we can't determine the better one between single

ratings data and single opinion scores data. Two conclusions can be drawn from results: opinions extracted

from the reviews is necessary in recommender systems, and also it's one of the important factors which could

Table 3. Performance of compared methods and proposed approach at rank 10.

Dataset	Yelp		Amazon	
Metrics	HR@10	NDCG@10	HR@10	NDCG@10
ItemPop	0.1218	0.0814	0.0906	0.0512
ItemKNN	0.4140	0.0768	0.3073	0.1706
PureSVD	0.4128	0.2762	0.3082	0.1701
Proposed framework	0.5994	0.3813	0.4352	0.2578

improve the performance; Moreover, ratings data is as important as opinion scores.

4.5.2 Result of proposed framework

When we adjust the parameters, we should not only pay attention to the best effect, but also avoid overfitting and underfitting. When the layers of MLP is {32, 16, 8}, blending ratio is 0.6 and prediction factors is 32, we obtain the optimal result based on Yelp dataset, HR = 0.5994 and NDCG = 0.3813; Under the default values of parameters, we get the best result based on Amazon dataset, the result is HR = 0.4352 and NDCG = 0.2578. Of course, we compare proposed framework under the best experiment configurations with methods mentioned in baselines. Moreover, we set the size of ranked list to 10. All of the results are showed in Table 3.

Based on two public datasets, the performance of proposed framework in this paper is better than all the methods in baselines, ItemPop performs worst, this simple method can't satisfy multiple requirements of the users, personalized becomes the basic criteria of recommender systems. PureSVD and ItemKNN provide almost the same recommendation effect, the difference of two methods is adaptability to data sparsity. ItemKNN performs better than PureSVD on Yelp dataset, but weak on the Amazon. There is a definite possibility that the Amazon dataset is sparser than Yelp. In this case, there are insufficient data for ItemKNN. Our proposed framework combines GMF and MLP, and the performance of this hybrid method better than GMF and MLP, for the sake of brevity, we omit this part of demonstration.

In summary, the experimental results show that our proposed framework achieves the best performance on two public datasets evaluated by both metrics, indicating the improvement based on two limitations do have an effect.

5 Conclusions

In this paper, we presented a framework combining opinion mining and neural networks, it not only alleviated data sparsity but also broke through the limit of traditional collaborative filtering algorithm. To enhance ratings, we combined opinions extracted from reviews with users' ratings. The proposed recommender method is personalized and extensible.

Certainly, there are still some issues to be solved. We aimed to emphasize the importance of review elements and actual ratings in recommender systems, the technique of opinion mining we adopted was basic, opinions data existed bias and noise. We will embed advanced opinion mining methods in recommender systems in future work.

Moreover, after becoming aware of the importance of deep neural networks in recommender systems, we hope to further explore the usage of deep neural networks in rating prediction.

References

1. Y. Koren, R. Bell, *Recommender Systems Handbook*, 145-186 (2015)
2. X. He, L. Liao, H. Zhang, L. Nie, X. Hu, WWW '17, 173-182 (2017)
3. L. Chen, G. Chen, F. Wang, UMUAI, **25**, 99-154 (2015)
4. D. Poirier, F. Fessant, I. Tellier, WI-IAT '10, **1**, 204-207 (2010)
5. G. Ganu, N. Elhadad, A. Marian, WebDB, **9**, 1-6 (2009)
6. CC. Musat, Y. Liang, B. Faltings, IJCAI '13, 2684-2690 (2013)
7. S. Raghavan, S. Gunasekar, J. Ghosh, RecSys '12, 123-130 (2012)
8. W. Zhang, G. Ding, L. Chen, C. Li, C. Zhang, TIST, **4**, 1-17 (2013)
9. Štefan. Pero, T. Horváth, *Lecture Notes in Computer Science*, **7899**, 1-13 (2013)
10. Liu B, *Handbook of Natural Language Processing*, **1**, 1-38 (2010)
11. L. K. Hansen, A. Arvidsson, F. A. Nielsen, CCIS, **185**, 34-43 (2011)
12. K. He, X. Zhang, S. Ren, and J. Sun, CVPR, 770-778 (2016)
13. R. Collobert, J. Weston, ICML, 160-167 (2008)
14. S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme, UAI '09, 452-461 (2009)
15. A. M. Elkahky, Y. Song, X. He, WWW '15, 278-288 (2015)
16. X. He, T. Chen, M. -Y. Kan, X. Chen, CIKM '15, 1661-1670 (2015)

17. P. Cremonesi, Y. Koren, R. Turrin, RecSys '10, 39-
B. Sarwar, G. Karypis, J. Konstan, WWW '01, **4**,
285-295 (2001)