Exploiting the Sentimental Bias between Ratings and Reviews for Enhancing Recommendation

Yuanbo Xu[1], Yongjian Yang[1], Jiayu Han[1], En Wang[1,*], Fuzhen Zhuang[2,3], Hui Xiong[4]
[1]Department of Computer Science and Technology, Jilin University, Changchun 130012, China;
{yuanbox15, jyhan15}@mails.jlu.edu.cn, {yyj, wangen}@jlu.edu.cn.
[2]Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS),
Institute of Computing Technology, CAS, Beijing 100190, China; zhuangfuzhen@ict.ac.cn.
[3]Univeristy of Chinese Academy of Science, Beijing 100049, China
[4]MSIS Department, Rutgers University; hxiong@rutgers.edu.

***Abstract*—In real-world recommendation scenarios, there are two common phenomena: *1)* users only provide ratings but there is no review comment. As a result, the historical transaction data available for recommender system are usually unbalanced and sparse; *2)* Users' opinions can be better grasped in their reviews than ratings. This indicates that there is always a bias between ratings and reviews. Therefore, it is important that users' ratings and reviews should be mutually reinforced to grasp the users' true opinions. To this end, in this paper, we develop an opinion mining model based on convolutional neural networks for enhancing recommendation (NeuO). Specifically, we exploit a two-step training neural networks, which utilize both reviews and ratings to grasp users' true opinions in unbalanced data. Moreover, we propose a *S*entiment *C*lassification scoring method (*SC*), which employs dual attention vectors to predict the users' sentiment scores of their reviews. A combination function is designed to use the results of SC and user-item rating matrix to catch the opinion bias. Finally, a *M*ultilayer perceptron based *M*atrix *F*actorization (*MMF*) method is proposed to make recommendations with the enhanced user-item matrix. Extensive experiments on real-world data demonstrate that our approach can achieve a superior performance over state-of-the-art baselines on real-world datasets.**

***Keywords*-Opinion bias, recommender systems, convolutional neural network, dual attention vectors**

I. Introduction

Recommender system is an elaborate and well-designed system, which is widely applied to e-commerce websites (Amazon, Taobao and Netflix) [1, 2]. Basically, a conventional recommender system utilizes the known information, such as users' attributes, browsing history, purchasing history, ratings and reviews to profile their preferences on different unconsumed items, then it makes an accurate recommendation [3].

However, we find two common but interesting phenomena: *1)* users prefer to rate their orders rather than review them after purchasing. *2)* users' opinions are usually not indicated by their reviews or ratings separately (as shown in Fig.1). In this paper, we focus on how to exploit users' true opinions on their consumed items to tackle the unbalanced dataset and opinion bias problems. Based on two observed phenomena, we design ***Neu***ral-network based ***O***pinion mining model (***NeuO***). NeuO consists of two modules: sentiment classification scoring (SC) module and MLP matrix factorization recommendation (MMF) module. For the comments with ratings only, we treat them as users' true opinions and input them into MMF directly. While for the comments with both ratings and reviews, we feed the reviews into SC, calculate the opinion bias on ratings to achieve users' true opinions. And then we feed them into MMF. NeuO is a two-step training neural network framework and is quite easy to tune. But it is actually effective to tackle the problems above and can be applied in many real-world scenarios.

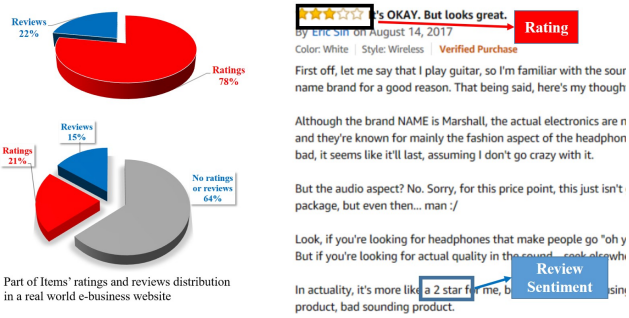


Figure 1. Left figure shows statistics in a real-world dataset: More than half users do not rate or review; among the users who give comments, 78% only rate and 22% give reviews. The dataset is unbalanced. The right figure is an example which explicitly shows the opinion bias between ratings and reviews, also the trigger of this research. However, it's difficult to catch the bias from innumerable users and reviews in most situations.

Our contributions can be summarized as follows:

- We propose NeuO, which is effective to tackle unbalanced and sparse data and catch the opinion bias from reviews to ratings explicitly.
- We propose a novel neural sentiment analysis method: Sentiment Classification Score (SC) to calculate users'

* Corresponding author

978-1-5386-9159-5/18/$31.00 ©2018 IEEE
DOI 10.1109/ICDM.2018.00185

sentiment scores with reviews. Dual attention vectors are applied to SC for improving the performance. Moreover, we design a Combination Function to catch the opinion bias explicitly.

- We conduct extensive experiments on real-world datasets, in which the encouraging results demonstrate that our proposed method tackles the unbalanced data in a uniform framework with a stable performance and obtains lower errors than state-of-the-art recommender system baselines.

II. Details of NeuO

A. Sentiment Classification scoring

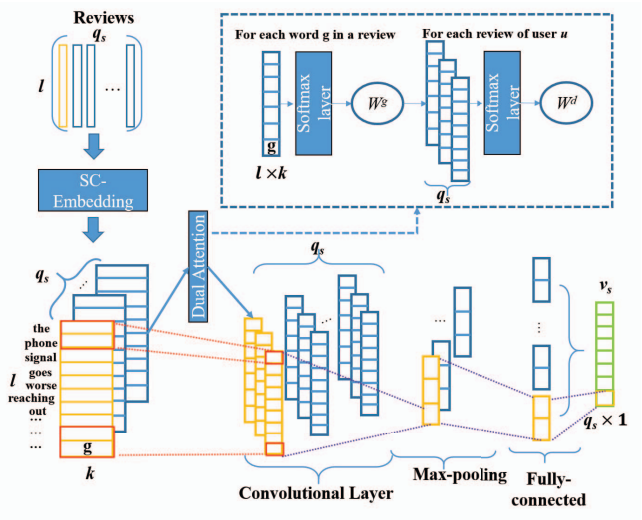


Figure 2. Sentiment classification scoring module

We build SC as shown in Fig.2. let U, I be the sets of users and items respectively, $|U| = m, |I| = n$. User-item matrix $R \in \mathbb{R}^{n \times m}$ consists of m users, n items. And r_{ui} stands for the user u's rating on item i. V stands for the whole review text.

The input of SC is V, and the output is a set of vectors V_s. We employ $\mathbb{F}_\alpha$ as the structure of SC, and α stands for the parameters in SC:

$$V_s = f^{out}(f^Q(f^{q-1}(...f^2(f^1(V))))). \tag{1}$$

In our proposed model, there are four hidden layers ($Q = 4$). The first layer is an embedding layer, which attempts to transfer users' reviews V into a set of dense feature vectors. We arrange all the reviews of user u, $v_u \in V$ to be a matrix $R_s \in \mathbb{R}^{l \times u_s}$. For each R_s, the embedding layer can be expressed as follows:

$$f^1 : V \to R_s \to \mathbb{R}^{l \times q_s \times k}, \tag{2}$$

where q_s equals to the number of items that the user u have rated, l stands for the length of reviews and k stands for the latent dimension for embedding words. We employ Word2Vec [11], which is a mature tool to do this word embedding.

After word embedding, each word in reviews is transferred into a k-dimension vector, and the latent description of user u's reviews are as follows:

$$D_u = d_1 \oplus d_2 \oplus d_3 ... \oplus d_{u_s}, \tag{3}$$

where d_i is an l-dimension vector whose entries are k-dimension latent vector for each word. d_i stands for the review of u on item i. $\oplus$ means the concatenation operator which merges the vectors to a $q_s \times l \times k$ description matrix D_u.

Then we feed d_u into the second layer, the convolutional layer. We utilize attention theory to adapt the different weight for each word in a review and each review in the review description matrix, both of which are named dual Attention Factors. In our model, we first add one attention factor W^g for each word embedding g in each review of user u:

$$\begin{aligned} W^g d_i &= w_1^g g_1 \oplus w_2^g g_2 \oplus w_3^g g_3 ... \oplus w_l^d g_l, \\ W^g &= \{ w^g | \sum_{i=1}^{l} w_i^g = 1 \}, \end{aligned} \tag{4}$$

where $w_1^g g_1$ means the element-wise production of w_1^g and g_1. Also another attention vector is applied for each review in review matrix:

$$\begin{aligned} W^d D_u &= w_1^d d_1 \oplus w_2^d d_2 \oplus w_3^d d_3 ... \oplus w_{q_s}^d d_{q_s}, \\ W^d &= \{ w^d | \sum_{i=1}^{q_s} w_i^d = 1 \}, \end{aligned} \tag{5}$$

where g stands for the k-dimention embedding for each word. W^g, W^d stand for the attention vector for each word in review and each review in review matrix. In this work, we utilize a softmax layer to gain weights of dual attention vectors, as shown in Fig.2.

Next, we feed output vectors obtained above into the convolutional layer to extract contextual features among words. It can be formulated as:

$$f^2 : C_u = f^2(W^2(W^d(W^g d)) + b^2), \tag{6}$$

where W^2, b^2 stand for the weights of filter functions in convolutional neural network. f^2 is a non-linear activation function (in this paper we employ $ReLU(x) = max(0, x)$). Moreover, we employ pl multiple filter functions (in this paper pl equals 3) to extract features and one filter function shares the same parameter.

The third layer is a max-pooling layer which extracts the most important information among many contextual features. The max-pooling function is defined as follow:

$$f^3 : C_u^m = max(C_{u1}, C_{u2}, ...C_{upl}). \tag{7}$$

The above vector will be parsed to the last layer: fully-connected layer:

$$f^4 : v_s = f^4(W^4 C_u^m + b^4), \tag{8}$$

where W^4, b^4 stand for the weights in full-connected neural network. f^4 can be a non-linear activation function or linear one. As we want to train the network with ratings instead of binary sentiment, we utilize softmax as the output function. The process of SC can be descried as follows:

$$v_s = SC(V\,|\alpha) \tag{9}$$

where α stands for $(W^g, W^d, W^2, W^4, b^2, b^4)$.

B. Combination Function

The process of combination function is shown in Fig.3:

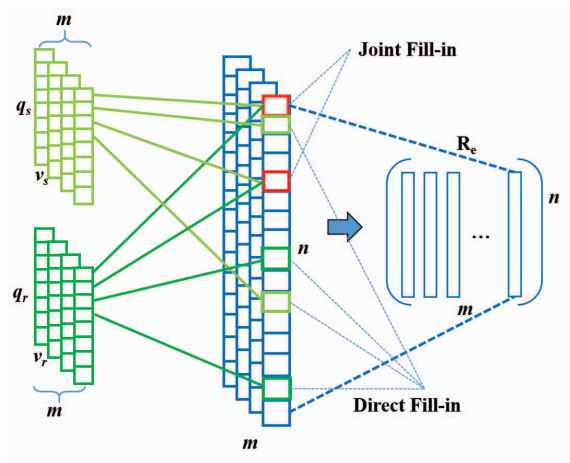


Figure 3. Combination Function module

After SC, we achieve a sentiment vector v_s for each user. Also we can extract each column from user-item rating matrix $R \in \mathbb{R}^{n\times m}$ as a rating vector v_r. Then we use a simple-but-effective function to build a joint vector v_e with v_r and v_s:

$$v_e = v_s \otimes v_r, \tag{10}$$

where $\otimes$ denotes the combination function of NeuO. Note that v_e is an n-dimension vector, v_s is a q_s-dimension vector whose entry is s_{ui} and v_r is a q_r-dimension vector whose entry is r_{ui}, where q_r equals to the item number that user has rated. Usually in real scenarios, $q_r \geq q_s$. Without the loss of generality, we just assume that $q_r \neq q_s$.

No matter we use review-based or rating-based recommender system, some important information hidden in the other side will be ignored. And it's difficult for both methods to tackle unbalanced data. Meanwhile, some users may make some useless reviews for some reasons (malicious reviews, rush comment, etc.). If the user rated an item and wrote a review, we can compute the sentiment score s_{ui} from SC and get a rating r_{ui} from R. The gap between them is exactly the opinion bias Ob we want to detect. If $Ob_{ui} = |s_{ui} - r_{ui}|$ is too large, we treat the review as a bad review and drop it. If Ob of a user is too large and too often, we treat the user as a bad user and drop him. Basically, we want to enhance the expression ability with reviews and ratings and filter the useless reviews at the same time. So the combination function $\otimes$ can be summarized as following three situations:

- **Direct Fill-in**: If the user u only rated or reviewed the item i, $\otimes$ fills v_e with either r_{ui} or s_{ui}.
- **Joint Fill-in**: If the user u rated and reviewed the item i, and did not be dropped, $\otimes$ fills v_e with fixed weighted linear function: $v_{ei} = \varepsilon s_{ui} + (1-\varepsilon) r_{ui}, \varepsilon \in (0,1)$.
- **Drop**: If $Ob_{ui} \geq \mu$, $\otimes$ drops the user' sentiment score s_{ui} for item i; if $(drop_u/q_s) \geq \iota$, $\otimes$ drops user u as a bad user. μ, ι are the thresholds predefined, and $drop_u$ is the number of u's dropped reviews.

It's obvious that with this $\otimes$, we can utilize both ratings and reviews to relieve the unbalance and sparsity of datasets, and catch the opinion bias explicitly. The process of Combination Function can be described as follows:

$$R_e = CB(v_s, v_r\,|\beta) \tag{11}$$

where β denotes $(\varepsilon, \mu, \iota)$.

C. MLP-based Matrix Factorization

We build a MLP-based matrix factorization (MMF) to make recommendations with R_e.

MMF module is composed of an embedding layer and MLP neural network. We build MMF as shown in Fig.4.

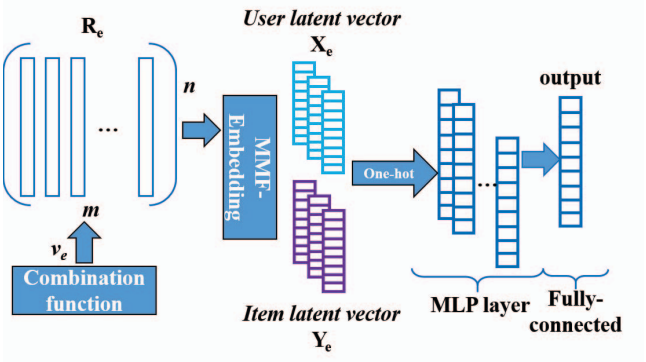


Figure 4. MLP-based Matrix Factorization module

Note that R_e has the same size as original matirx R. Then we utilize weighted Matrix Factorization in embedding layer to achieve user latent vectors and item latent vectors separately. MMF employs Matrix Factorization to embed R_e into a p-dimensional latent space. We introduce τ as parameters to represent the multiple possible factorization results, like parameterized Matrix Factorization (pMF). Given an enhanced rating matrix R_e, and a sequence of parameters,

$\tau = \{\tau_1, \tau_2, ..., \tau_p\}$, users and items can be mapped into latent vectors X_e and Y_e:

$$R_e \approx (X_e)^T \Sigma_e Y_e; X_e \in \mathbb{R}^{p \times n}, Y_e \in \mathbb{R}^{p \times m}, \quad (12)$$

where Σ_e is a $p \times n$ diagonal matrix whose entries are τ. After the embedding users and items, we use one-hot encoding to control the inputs of neural network, where z_u and z_i are the input control vectors. We express the predictions of ratings as follows:

$$\tilde{r}_{ui} = h(X_e z_u, Y_e z_i, |\tau, \gamma), \quad (13)$$

where $X \in \mathbb{R}^{p \times n}$, $Y \in \mathbb{R}^{p \times m}$. τ denotes the parameter of MMF-embedding layer and γ denotes other parameters in MMF. Different τ controls the different embedding results.

Existing methods often employ element-wise products to predict ratings, which is suitable to measure the accuracy problem. However, our enhanced rating matrix contains the information of reviews and ratings, which makes it more complex than a traditional matrix factorization problem. Element-wise products cannot measure the complex relationships in recommender systems [13]. So we employ direct vector concatenation instead of element-wise products to feed $[X, Y]$=$[X_e z_u, Y_e z_i]$ into an MLP network $\mathbb{H}_\gamma$. The t-th hidden layer is denoted as h^t, which is a non-linear function of former hidden layer h^{t-1}. And $\mathbb{H}$ employs $ReLU(x)$=$max(0, x)$ as activation function:

$$h^t(X, Y) = ReLU(W^t h^{t-1}(X, Y) + b^t), \quad (14)$$

where w^t and b^t are the parameters of the t-th hidden layer. Above all, we get the formulation of MMF:

$$\tilde{r} = h^{out}(h^{T-1}(h^{t-1}(...h^2(h^1(X, Y))))), \quad (15)$$

where T is the number of hidden layer of tower neural structure. In this paper, we build a 64-16-32-8 MLP (T=4). As we are predicting the ratings, on the top of hidden layers, we use softmax activation function as the output layer h^{out}.

The process of MMF can be described as follows:

$$\tilde{r} = MMF(R_e |\tau, \gamma), \quad (16)$$

where γ stands for the parameters in MLP of MMF. After MMF, we can get latent vectors $\tilde{X}_e, \tilde{Y}_e$. The ratings of unrated items $\tilde{r}_{ui}$ for each user can be calculated by Eq. 13. According to $\tilde{r}_{ui}$, we could recommend Top-k rating items to users.

III. Experiments

A. Dataset

We conduct extensive experiments on Amazon.com dataset[1] and Yelp for RecSys[2]. We also collect a real-world dataset from Taobao[3] to achieve this goal. All the datasets contain rating range from 1 to 5, and we use 5-cross validation to divide the datasets, with 80% as training set, 10% as test set and 10% as validation set. The details of datasets are summarized in Table I.

[1]https://jmcauley.ucsd.edu/data/amazon

[2]https://www.kaggle.com/c/yelp-recsys-2013

[3]https://www.taobao.com

Table I
THE CHARACTERISTICS OF DATASETS

Dataset	Amazon	Yelp	Taobao
#user	30,759	45,980	10,121
#item	16,515	11,537	9,892
#review	285,644	229,900	10,791
#rating	285,644	229,900	49,053
Sparsity	0.051%	0.043%	0.049%
Avg # words per review	104	130	89
Avg # reviews per user	9.29	5.00	1.06

B. Baselines

We compare our model with the following baselines:

1) Attentive Collaborative Filtering (A-CF) [14], 2) Adaptive Matrix Factorization (A-MF) [15], 3) Text-driven Latent Factor Model (TLFM) [16], 4) Convolutional matrix factorization (ConvMF+).

C. Experimental Results and Discussions

1) Recommendation Accuracy: We employ Mean Squared Error (MSE) and Hitting Rate (HR) as the metrics for NeuO as well as other baselines. MSE results are shown in Table II, and HR results are shown in Table III.

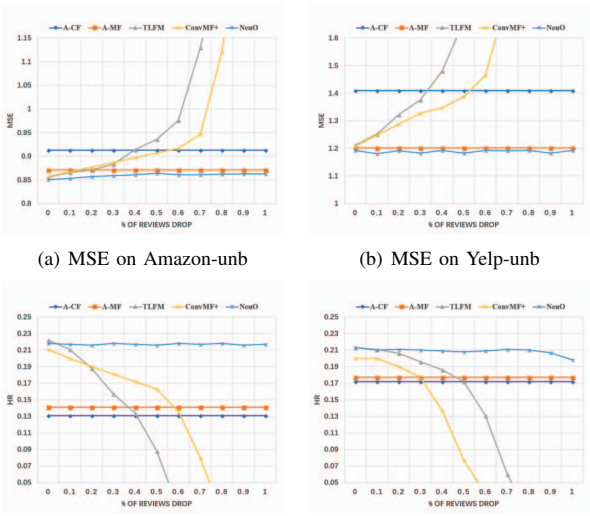
Table II
RATING PREDICTION FOR DIFFERENT MODELS (MSE)

Dataset	A-CF	A-MF	TLFM	ConvMF+	NeuO
Amazon	0.913	0.871	0.856	0.857	**0.851**
Yelp	1.410	1.201	1.211	1.208	**1.193**
Taobao	1.512	1.341	1.674	1.222	**1.195**

Table III
TOP-5 FOR DIFFERENT MODELS (HR)

Dataset	A-CF	A-MF	TLFM	ConvMF+	NeuO
Amazon	0.131	0.141	**0.222**	0.211	0.218
Yelp	0.172	0.177	**0.213**	0.200	**0.213**
Taobao	0.091	0.089	0.049	0.100	**0.185**

MSE result clearly proves the effectiveness of our model. A-CF and A-MF perform worse than other baselines with the limit of data sparsity and unbalance. Note that TLFM and ConvMF+ can achieve the same level performance as NeuO on Amazon and Yelp. However, when applied to unbalanced Taobao dataset, our proposed model overperform all the baselines with a large gap. The reason is that NeuO is designed with the consideration of unbalanced data in real-world applications. And the Combination Function can utilize both reviews and ratings, which enhance the expression ability of original datasets.



(a) MSE on Amazon-unb (b) MSE on Yelp-unb

(c) HR on Amazon-unb (d) HR on Yelp-unb

Figure 5. Robustness for Unbalanced dataset with different models

2) Robustness for Unbalanced Data: From the comparisons above, we can clearly see that although TLFM and ConvMF+ can achieve the same level performance on pre-filtered datasets Amazon and Yelp (shown in Table II, Table III), when the datasets are unbalanced (some reviews are dropped to simulate the real scenarios), their performance declines rapidly. Note that when we drop all the reviews, TLFM and ConvMF+ perform nearly as random methods. A-MF and A-CF are rating-based methods, so their performance is relatively stable but not surprising. However, our model can tackle both reviews and ratings. When the data becomes unbalanced, NeuO can utilize additional information from the other side (ratings) to build the enhanced user-item matrix, which can greatly improve the robustness. Moreover, when the reviews are totally dropped, NeuO can be treated as a rating-based recommender system and also achieve a stable performance like A-CF and A-MF.

3) Effect of Opinion Bias: We feed enhanced matrix R_e and original R separately into our method NeuO and two rating-based baselines: A-MF and A-CF to see the effect of opinion bias of NeuO.

Table IV
EFFECT OF OPINION BIAS

	Amazon-unb		Yelp-unb		Taobao	
MSE	R	R_e	R	R_e	R	R_e
A-CF	0.973	**0.942**	1.41	**1.31**	1.512	**1.342**
A-MF	0.873	**0.852**	1.201	**1.194**	1.341	**1.197**
MMF	0.875	**0.851**	1.204	**1.193**	1.356	**1.195**
HR						
A-CF	0.131	**0.170**	0.172	**0.200**	0.091	**0.161**
A-MF	0.141	**0.176**	0.177	**0.205**	0.089	**0.156**
MMF	0.150	**0.218**	0.174	**0.213**	0.094	**0.185**

The results in Table IV clearly confirm the effect of opinion bias. All the methods perform better (10% with A-CF, 15% with A-MF and 17% with MMF) with R_e than R. The reasons are 1) Combination Function utilizes both ratings and reviews, which enriches the information of original rating matrix and make the ratings more accurate to the true opinions of users. 2) NeuO increases the density of the matrix, which benefits the MF-based methods most (A-MF and MMF).

Moreover, we show a potential application of opinion bias on real-world dataset Taobao. There are always complex scenarios in a real e-website: some bad users try to make profits from websites by giving a bad review on purpose. They may write some non-relative comments but give an item a very low rating, or give a good rating but bad comments. We focus on the users dropped by combination function of NeuO and treat the process as a malicious user detection. Among all 10,121 users collected by us, NeuO drops 7 users and 142 reviews. Some desensitization information of these users are shown in Table V:

Table V
STATISTICS OF DROPPED USERS

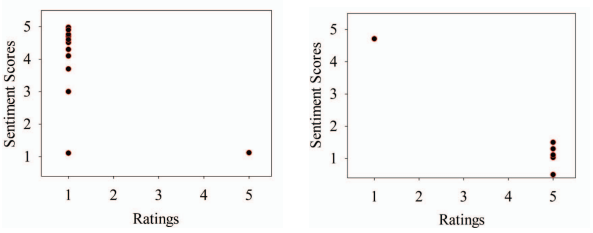
	ratings	reviews	5 star	4 star	3 star	2 star	1 star
user1	23	17	3	0	0	0	20
user2	46	22	4	1	0	0	41
user3	17	11	0	0	0	0	17
user4	22	22	3	0	0	0	22
user5	16	12	1	0	0	0	15
user6	5	2	0	0	0	0	5
user7	9	9	8	0	0	0	1

All those users rate a lot bad reviews and ratings among all the reviews and ratings (except $user7$, which is another type of malicious user). Moreover, we use $user5$ and $user7$ as examples, by plotting the sentiment vector v_s and rating vector v_r:

From Fig.6, we can see that $user5$ is the one who always gives a good review but a low rating (almost get sentiment score 5 for all 1-star ratings) and $user7$ is the one who always gives a bad review but a high rating. According to the opinion bias catched by NeuO, we can confirm that both two users are abnormal for e-commerce websites and may be malicious users. It shows the potency of NeuO to be applied to some special applications in real scenarios.

IV. CONCLUSION

In this paper, we proposed NeuO, which focused on capturing opinion bias between review content and users' ratings in unbalanced datasets. The experimental results on Amazon, Yelp and Taobao datasets showed that our method outperforms state-of-the-art baselines in accuracy, achieves a stable performance in unbalanced datasets.and able to catch the opinion bias correctly



(a) Ratings and senti-score of $user5$ (b) Ratings and senti-score of $user7$

Figure 6. Reason for malicious user detection

V. Acknowledge

this paper is supported by the National Natural Science Foundation of China under Grant No. 61773361, 61473273, 61772230. And Natural Science Foundation of China for Young Scholars No. 61702215, China Postdoctoral Science Foundations No. 2017M611322 and No. 2018T110247. Project of Special Fund for Industrial Innovation in Jilin No. 2017C032-1. Jilin Technology Development Project No.20160204021GX.

References

[1] J. Lu, D. Wu, M. Mao, W. Wang, and G. Zhang, "Recommender system application developments: a survey," *Decision Support Systems*, vol. 74, pp. 12–32, 2015.

[2] J. Yang, C. Liu, M. Teng, J. Chen, and H. Xiong, "A unified view of social and temporal modeling for b2b marketing campaign recommendation," *IEEE Transactions on Knowledge and Data Engineering*, 2017.

[3] C. He, D. Parra, and K. Verbert, "Interactive recommender systems: A survey of the state of the art and future research challenges and opportunities," *Expert Systems with Applications*, vol. 56, pp. 9–27, 2016.

[4] X. Yang, Y. Guo, Y. Liu, and H. Steck, "A survey of collaborative filtering based social recommender systems," *Computer Communications*, vol. 41, pp. 1–10, 2014.

[5] G. Adomavicius and A. Tuzhilin, "Context-aware recommender systems," in *Recommender systems handbook*. Springer, 2015, pp. 191–226.

[6] J. Beel, B. Gipp, S. Langer, and C. Breitinger, "paper recommender systems: a literature survey," *International Journal on Digital Libraries*, vol. 17, no. 4, pp. 305–338, 2016.

[7] N. Rubens, M. Elahi, M. Sugiyama, and D. Kaplan, "Active learning in recommender systems," in *Recommender systems handbook*. Springer, 2015, pp. 809–846.

[8] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W.-Y. Ma, "Collaborative knowledge base embedding for recommender systems," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 2016, pp. 353–362.

[9] Z. D. Champiri, S. R. Shahamiri, and S. S. B. Salim, "A systematic review of scholar context-aware recommender systems," *Expert Systems with Applications*, vol. 42, no. 3, pp. 1743–1758, 2015.

[10] U. Panniello, A. Tuzhilin, and M. Gorgoglione, "Comparing context-aware recommender systems in terms of accuracy and diversity," *User Modeling and User-Adapted Interaction*, vol. 24, no. 1-2, pp. 35–65, 2014.

[11] B. Xue, C. Fu, and Z. Shaobin, "A study on sentiment computing and classification of sina weibo with word2vec," in *Big Data (BigData Congress), 2014 IEEE International Congress on*. IEEE, 2014, pp. 358–363.

[12] S. Seo, J. Huang, H. Yang, and Y. Liu, "Interpretable convolutional neural networks with dual local and global attention for review rating prediction," in *Proceedings of the Eleventh ACM Conference on Recommender Systems*. ACM, 2017, pp. 297–305.

[13] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 173–182.

[14] J. Chen, H. Zhang, X. He, L. Nie, W. Liu, and T.-S. Chua, "Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention," in *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 2017, pp. 335–344.

[15] Y. Ning, Y. Shi, L. Hong, H. Rangwala, and N. Ramakrishnan, "A gradient-based adaptive learning framework for efficient personal recommendation," in *Proceedings of the Eleventh ACM Conference on Recommender Systems*. ACM, 2017, pp. 23–31.

[16] K. Song, W. Gao, S. Feng, D. Wang, K.-F. Wong, and C. Zhang, "Recommendation vs sentiment analysis: a text-driven latent factor model for rating prediction with cold-start awareness," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. AAAI Press, 2017, pp. 2744–2750.

[17] D. Kim, C. Park, J. Oh, S. Lee, and H. Yu, "Convolutional matrix factorization for document context-aware recommendation," in *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 2016, pp. 233–240.