Neural Networks 111 (2019) 77–88

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

NeuO: Exploiting the sentimental bias between ratings and reviews with neural networks

Yuanbo Xu[a], Yongjian Yang[a], Jiayu Han[a], En Wang[a,*], Fuzhen Zhuang[b,c], Jingyuan Yang[d], Hui Xiong[e]

[a] *Jilin University, Changchun, Jilin, China*
[b] *Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS), Institute of Computing Technology, CAS, Beijing 100190, China*
[c] *University of Chinese Academy of Sciences, Beijing 100049, China*
[d] *George Mason University, VA, USA*
[e] *Rutgers, The State University of New Jersey, NJ, USA*

ARTICLE INFO

Article history:
Received 30 September 2018
Received in revised form 7 November 2018
Accepted 14 December 2018
Available online 8 January 2019

Keywords:
Opinion bias
Recommender systems
Convolutional neural network
Dual attention vectors

ABSTRACT

Traditional recommender systems rely on user profiling based on either user ratings or reviews through bi-sentimental analysis. However, in real-world scenarios, there are two common phenomena: *(1)* users only provide ratings for items but without detailed review comments. As a result, the historical transaction data available for recommender systems are usually unbalanced and sparse; *(2)* in many cases, users' opinions can be better grasped in their reviews than ratings. For the reason that there is always a bias between ratings and reviews, it is really 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. Specifically, we exploit 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 (***SC***) method, which employs dual attention vectors to predict the users' sentiment scores of their reviews rather than using bi-sentiment analysis. Next, a combination function is designed to use the results of SC and user–item rating matrix to catch the opinion bias. It can filter the reviews and users, and build an enhanced user–item matrix. 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 several real-world datasets (Yelp, Amazon, Taobao and Jingdong) demonstrate that (1) our approach can achieve a superior performance over state-of-the-art baselines; (2) our approach is able to tackle unbalanced data and achieve stable performances.

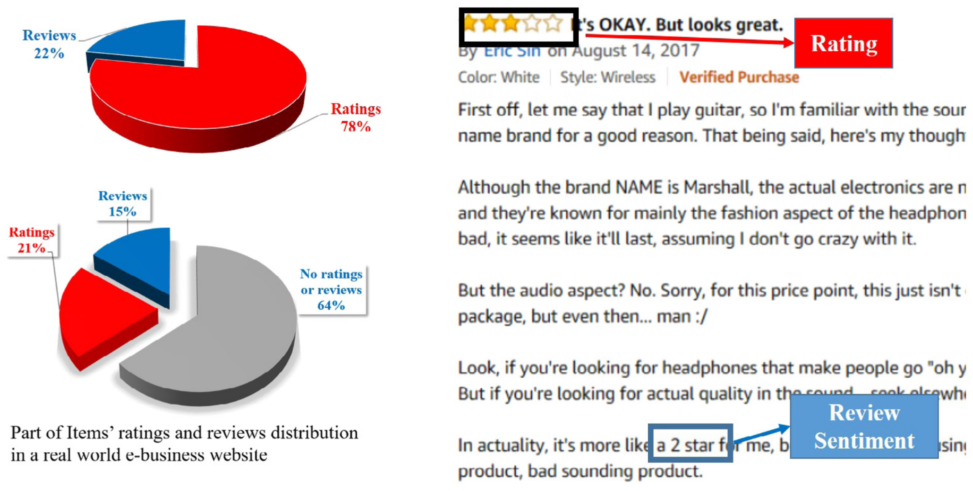
1. Introduction

Recommender system is an elaborate and well-designed system, which is widely applied to e-commerce websites (Amazon, Taobao and Netflix) (Lu, Wu, Mao, Wang, & Zhang, 2015; Yang, Liu, Teng, Chen & Xiong, 2017). In general, 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 (He, Parra & Verbert, 2016). To exploit the users' opinions on consumed items, the businesses always use reviews and ratings, which are the most pervasive context utilized in recommender systems (Adomavicius & Tuzhilin, 2015; Yang, Guo, Liu, & Steck, 2014).

Because of the different data structures of users' ratings and reviews, it is still a challenge to utilize them both to make recommendations (Beel, Gipp, Langer, & Breitinger, 2016). Some researchers believe that users' opinions on different items are indicated by their ratings (from 0 to 5 in most situations), which are treated as the satisfaction extent (Rubens, Elahi, Sugiyama, & Kaplan, 2015; Zhang, Yuan, Lian, Xie, & Ma, 2016). While some researchers try to use reviews, which are marked to be positive or negative by sentiment classification methods to make recommendations (Champiri, Shahamiri, & Salim, 2015; Panniello, Tuzhilin, & Gorgoglione, 2014). Correspondingly, many sophisticated recommender systems have been designed to profile users' preferences with either their reviews or ratings. Indeed, both of them have achieved remarkable recommendation results when applied in e-commerce website datasets. Ideally, if we can get both users' reviews and ratings on their consumed items, it is natural that we may exploit users' comprehensive and true opinions to make a more accurate recommendation.

* Corresponding author.
E-mail address: wangen@jlu.edu.cn (E. Wang).

https://doi.org/10.1016/j.neunet.2018.12.011
0893-6080/© 2018 Elsevier Ltd. All rights reserved.



's OKAY. But looks great.
By Eric Sin on August 14, 2017
Color: White | Style: Wireless | Verified Purchase

First off, let me say that I play guitar, so I'm familiar with the sour
name brand for a good reason. That being said, here's my though

Although the brand NAME is Marshall, the actual electronics are n
and they're known for mainly the fashion aspect of the headphon
bad, it seems like it'll last, assuming I don't go crazy with it.

But the audio aspect? No. Sorry, for this price point, this just isn't
package, but even then... man :/

Look, if you're looking for headphones that make people go "oh y
But if you're looking for actual quality in the sound... seek elsewh

In actuality, it's more like a 2 star for me, b sin
product, bad sounding product.

Fig. 1. The left figure shows statistics in a real-world dataset: more than half users do not rate or review; among the users who give feedback, 78% only rate items and 22% give both reviews and ratings. Therefore, the dataset contains unbalanced number of ratings and reviews. The right figure is an example which explicitly shows the opinion bias between ratings and reviews, and also the motivation of this research. However, it is difficult to catch the bias from innumerable users and reviews in most situations.

However, after investigating reviews and ratings in real-world datasets, we find two common but interesting phenomena: (1) users prefer to rate their orders rather than review them after purchasing, which means that we cannot get both reviews and ratings at the same time. It is quite often the case with Taobao and Jingdong, which makes the data unbalanced and sparse. (2) users' true opinions could not be directly reflected by either their reviews or ratings separately (as shown in Fig. 1). In other words, rating-based or review-based recommendation systems may lose some important information on the other side and fail to exploit users' true opinions on consumed items. Moreover, as shown in Fig. 1, there is an opinion bias between ratings and reviews in some cases. If we ignore the opinion bias and combine reviews and ratings directly, we may infer some deviant opinions of users and make inaccurate recommendations. Therefore, it is necessary to design a recommender system which can utilize both ratings and reviews to exploit users' true opinions on different items, considering the opinion bias. Users' true opinions are defined as the users' unbiased preferences on different items based on the known information about the users (ratings and reviews).

Taking a deep insight into the above two phenomena, we find that users' true opinions on consumed items are affected by ratings and reviews mutually. Intuitively, ratings are the proper metric to evaluate users' opinions. And reviews can be treated as an opinion bias on ratings. However, the bias is very difficult to model because the effect of reviews is not stationary (Yang et al., 2014), especially when the dataset is unbalanced and sparse. In order to understand users' authentic opinions on items, it is a significant work to catch the opinion bias explicitly from ratings to reviews.

To address all the challenges and solve the opinion bias problem for real-world recommendations, we did some preliminary work.[1] We focused on how to exploit users' true opinions on their consumed items to tackle the unbalanced dataset and opinion bias challenges. Based on the above two observed phenomena, we proposed to design an elaborate neural-network based recommendation system, ***Neu***ral-network based ***O***pinion mining model (***NeuO***). Specifically, NeuO consisted of two modules: sentiment classification scoring (SC) module and MLP matrix factorization recommendation (MMF) module. For the comments with ratings only, we treated them as users' true opinions and put them into MMF directly. While for the comments with both ratings and reviews, we fed the reviews into SC, calculated the opinion bias on ratings to achieve users' true opinions, and then input them into MMF to make recommendations.

In this paper, we first further improve and introduce the framework of NeuO, and design an elaborate Combination Function to tackle the opinion bias problem. Also, we make more experiment on another self-collected real-world dataset (Jingdong) to validate our proposed model. Moreover, we add extensive experiments to prove the effectiveness of applying dual-attention vectors on opinion bias problem and exploit the potentialities of NeuO by deciding different parameters. To the best of our knowledge, it is the first work to combine reviews and ratings together to catch the opinion bias explicitly, in order to make more accurate recommendations. Except for the details in SC and MMF (we will discuss in following sections), basically, NeuO is a two-step training neural network framework and is quite easy to tune. However, it is indeed effective to tackle the above problems and can be widely applied in many real-world scenarios.

Our contributions can be summarized as follows:

- We propose an elaborate neural-network based recommendation framework: NeuO, which is effective to tackle unbalanced and sparse data and catch the opinion bias from reviews to ratings explicitly. Without loss of generality, this framework can cover some basic review-based or rating-based recommender systems.
- We propose a novel neural sentiment analysis method: Sentiment Classification Score (SC) to calculate users' sentiment scores with reviews. Dual attention vectors are applied to SC for improving the performance. Moreover, we design an elaborate Combination Function to catch the opinion bias explicitly. Finally, an MLP-based matrix factorization (MMF) method is also proposed to make recommendations with these true opinions.
- We conduct extensive experiments on four real-world datasets, in which the encouraging results demonstrate that our proposed method (1) tackles the unbalanced data in a uniform framework with a stable performance (2) obtains less errors than state-of-the-art recommender system baselines (3) is able to be applied in real-world scenarios.

The rest of the paper is organized as follows: in Section 2, we give a brief introduction to the related work. Basic definitions and problem definitions are given in Section 3. Then, we introduce the framework of our approach in Section 4 and the details in Section 5.

[1] Accepted as a short paper by the IEEE International Conference on Data Mining (ICDM) 2018.

In Section 6, we conduct experiments to evaluate our proposed methods. Finally, we conclude our work in Section 7.

2. Related work

2.1. Rating-based and review-based recommender systems

From data aspect, recommender systems can be categorized into rating-based and review-based recommender systems (He, Parra et al., 2016; Ning, Shi, Hong, Rangwala, & Ramakrishnan, 2017). In rating-based models, Collaborative Filtering (CF) is the most popular method (Koren & Bell, 2015; Wei, He, Chen, Zhou, & Tang, 2017; Yang, Lei, Liu & Chua, 2017; Zhuang, Luo, Yuan, Xie, & He, 2017a; Zhuang, Zhang, Qian, Shi, Xie, & He, 2017), which mainly employs Matrix Factorization (MF) (Chin, Zhuang, Juan, & Lin, 2015; He, Zhang, Kan & Chua, 2016) to make recommendations. Yang, Lei et al. (2017) elaborately integrated twofold sparse information in social network, and TrustMF they proposed could achieve state-of-the-art performance. Wei et al. (2017) started to consider the possibility of utilizing neural network and CF both to tackle the cold-start issue. Chin et al. (2015) proposed a fast parallel stochastic gradient method for matrix factorization, which can be used to accelerate recommendations on a shared memory system. He, Zhang et al. (2016) proposed a matrix factorization based CF model to tackle the implicit feedback recommendations. Basically, rating-based models attempt to measure the relationships between different users and items with the user–item rating matrix. It achieves a good performance when the dataset is relatively dense and small. When the dataset is huge, sparse and unbalanced, the rating-based models perform limited by cold-start issues and data sparsity in most cases (Koren & Bell, 2015).

In review-based models, researchers utilize different methods (text analysis (Rosenthal, Farra, & Nakov, 2017), sentiment classification (Han, Zuo, Liu, Xu, & Peng, 2016)) to infer users' preferences through their reviews and make recommendations (Champiri et al., 2015; Liu, Wu, & Wang, 2015; Panniello et al., 2014). Champiri et al. (2015) summarized some state-of-the-art methodologies and theories which were often employed in review-based models. Moreover, it pointed out the weaknesses that the review-based recommendations were somehow uninteresting and repeated. Panniello et al. (2014) investigated the trade-off between accuracy and diversity in review-based models. Some researchers also want to construct a novel model to tackle the reviews. Liu et al. (2015) tried to use tenser-factorization theory to optimize the context-aware recommender system. Although both review-based and rating-based models can achieve satisfying results in some scenarios, they are not designed for utilizing both reviews and ratings on unbalanced data in real-world.

2.2. Recommender system with neural network

The combination of recommender systems and the neural network is becoming a hot research trend (Bai, Wen, Zhang, & Zhao, 2017; Chen, Zhang, He, Nie, Liu, & Chua, 2017; He, Liao, Zhang, Nie, Hu, & Chua, 2017; Lian, Zhang, Xie, & Sun, 2017; Yang, Bai, Zhang, Yuan, & Han, 2017; Yang, Xu, Wang, Han & Yu, 2017; Ying, Zhuang, Zhang, Liu, Xu, Xie, Xiong, & Wu, 2018; Zhuang, Luo, Yuan, Xie, & He, 2017). Researchers attempt to utilize the non-linear activation functions in the neural network to measure the relationships between users and reviews. He et al. (2017) utilized Multilayer perceptron (MLP) to design a network NeuCF to tackle implicit feedback recommendation problems. NeuCF is a rating-based model which can cover basic MF and CF and also achieve state-of-the-art performance. Moreover, Bai et al. (2017) focused on the relations between neighbors and proposed a neural-network based recommender system. Also, some researchers try to combine neural models with traditional machine learning to make recommendations. Yang et al. (2017) combined semi-supervised and neural network, bridged them and reinforced mutually. Yang, Xu et al. (2017) proposed a novel concept: Serendipity and they utilized an MLP-based network to tackle the serendipity issues in recommender systems. Attention vectors are also employed by some researchers. Chen et al. (2017) embedded users and items with different attention layers and achieved a good performance. Seo, Huang, Yang, and Liu (2017) used local and global attention vectors to optimize user embedding in recommender systems. These models put more attention on the methodology of neural network itself rather than the applications in real scenarios, which also achieve a satisfying performance on various prefiltered datasets. These works have made improvement in accuracy and efficiency, however, they almost validate their on standard datasets like movielens (Harper & Konstan, 2016), or yelp (Luca & Zervas, 2016), and are not designed for unbalanced date with opinion bias problem in real world. Therefore, we propose NeuO in this paper, which is designed for tackling unbalanced date with opinion bias problem, utilizing CNN and MLP to model users' preferences.

3. Preliminary

In this section, we give some basic definitions and problem definitions.

3.1. Basic definition

In a recommender system, let U be a set of m users $U = \{u_1, u_2 \dots u_m\}$, and I be a set of n items $I = \{i_1, i_2 \dots i_n\}$. r_{ui} means the rating user u marked for item i. So we build a rating user–item matrix $R_{n\times m}$, whose elements are r_{ui} for rated items and *null* for unrated items. As the same situation, v_{ui} means the review user u marked for item i. So we build a review user–item matrix $V_{n\times m}$.

As we introduced before, there is always opinion bias between ratings and reviews, so we define opinion bias as follows:

Definition 1 (*Opinion Bias (Ob)*)**.** Given a user–item pair, the opinion bias Ob_{ui} is defined as the bias between rating r_{ui} and review v_{ui}.

However, as the review v_{ui} consists of text, we employ a sentiment scoring model (it will be introduced in the following sections) to transform v_{ui} into a sentiment score s_{ui}. So the opinion bias Ob_{ui} can be calculated by $Ob_{ui} = |s_{ui} - r_{ui}|$. In our proposed model, we also need to consider unbalanced situation, which is defined as follows:

Definition 2 (*Unbalanced Situation*)**.** Given user–item rating matrix R and review matrix V, the unbalanced situation means that the number of ratings is not equal to the number of reviews, $|R| \neq |V|$, which may harm the traditional rating-based recommendation models or review-based recommendation models.

3.2. Problem definition

With Definitions 1 and 2, we can define our problem as follows:

Problem Definition-Unbiased Unbalanced co-recommendation: Given a user–item rating matrix R, and review matrix V, an unbiased unbalanced co-recommendation model should have the ability to pick up the opinion bias from the unbalanced situation and utilize both reviews and ratings to make a superior recommendation.

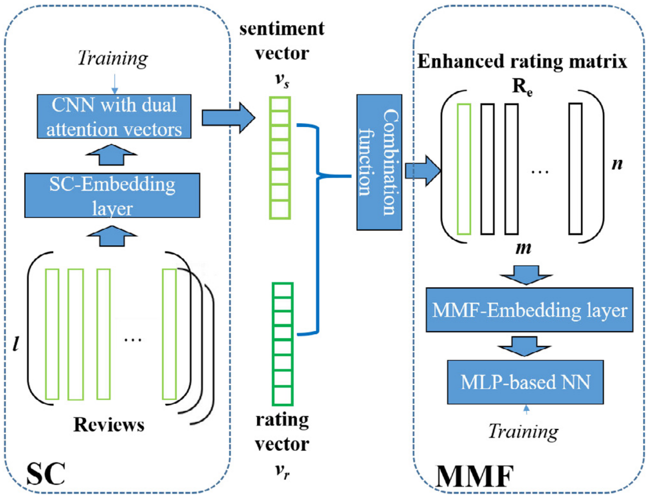


Fig. 2. The framework of NeuO.

4. The framework of NeuO

In this section, we briefly introduce the framework of Neural-network based Opinion mining model (NeuO). NeuO is composed of two modules: SC and MMF (Fig. 2). SC is a convolutional neural network for sentiment analysis. Different from the traditional bi-sentiment analysis, with feeding the reviews, SC outputs a sentiment score (ranging from 0 to 5) to fit the ratings instead of negative or positive. The output of SC for a user u is a sentiment vector v_s whose entry is the score for each item that u has reviewed. Meanwhile, we select the column in user–item rating matrix to form a rating vector v_r. By utilizing a Combination Function, we can get an enhanced rating matrix R_e. Then we feed R_e to MMF to achieve the accurate recommendations.

The advantage of NeuO is that it can tackle the unbalanced data that users' reviews and ratings are not matched. For two extreme scenarios: (1) we only have reviews in the training dataset. Usually, it is a traditional binary sentiment problem. NeuO can only utilize sentiment vector v_s to build R_e. And we could build a simple classifier to tackle the output of MMF because the output is a matrix whose entry ranges from 0 to 5. (2) We only have ratings in the training dataset. NeuO can replace R_e by original user–item matrix R. In this way, it can cover some basic matrix factorization methods in recommender systems. And NeuO has the ability to relieve the sparsity problem by building R_e through reviews and ratings together. In sum, NeuO could efficiently address the extreme situations.

Moreover, in real scenarios, we do not get into the above extreme situations frequently. Usually, we get some datasets that the number of users' ratings is more than that of the corresponding reviews' (the opposite situation almost could not happen in the real world). If we use review-based or rating-based methods separately, we may miss some important information hidden on the other side. For example, as described before, we cannot catch the opinion bias from reviews to ratings. NeuO combines sentiment vectors and rating vectors and reinforces both mutually. With the comparison between them, we can catch the bias explicitly and clearly. Also, the opinion bias can help us to understand users' preferences better and make more accurate recommendations.

Our proposed model NeuO is a two-step training process. The first step is sentiment scoring, which utilizes reviews as the input and the sentiment scores as the output. To train this CNN, we treat ratings as the ground truth. After pretraining the sentiment scoring model, NeuO feeds the data into this model, then utilizes Combination Function to catch the opinion bias and filter the data. Finally, NeuO utilizes MMF model to make recommendation. For a better understanding of NeuO, SC and Combination Function can be treated as a data preprocess and MMF model as a recommendation method. Most important notations are summarized in Table 1.

Table 1
Important notations.

Notation	Description
U	User set in our recommender system
I	Item set in our recommender system
m, n	Number of users/items
k	Latent embedding dimension in SC
p	Latent embedding dimension in MMF
l	length of reviews
r_{ui}	u's rating on item i
s_{ui}	u's senti-score on item i
R	User–item matrix with ratings r_{ui}
V	Reviews in recommender system
R_e	Enhanced rating matrix
v_r	Rating vector whose entry is r_{ui}
v_s	Sentiment vector whose entry is s_{ui}
q_r	Dimension of v_r
q_s	Dimension of v_s
α, β, γ	Parameters in SC, Combination function and MMF

5. Details of NeuO

In this section, we introduce sentiment classification (SC) scoring , Combination Function and MMF. SC is a convolutional neural network (CNN) with dual attention vectors for predicting sentiment score. Combination Function is applied to utilizing both reviews and ratings and building an enhanced rating matrix. Also it can catch the opinion bias explicitly. MMF is a Multilayer Perceptron (MLP) neural network to make recommendations.

5.1. Sentiment classification (SC) scoring

Sentiment classification (SC) scoring module is a convolutional neural network with dual attention vectors, which utilizes the reviews to predict sentiment scores (range from 0 to 5) instead of binary classification results. We build SC as shown in Fig. 3.

In our proposed recommender system, 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. The whole process can be expressed as follows:

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

There are four hidden layers ($Q = 4$) in our proposed model. The first layer is an **embedding layer**, which attempts to transfer users' reviews V into a set of dense feature vectors. It is an effective way to discover the relations among each word. 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:

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

where q_s is equal to the number of items that the user u has rated, l stands for the length of reviews and k stands for the latent dimension for embedding words. We employ Word2Vec (Xue, Fu, & Shaobin, 2014), which is a sophisticated tool for word embedding.

Note that if the length of review is longer than l, we reserve l words and then drop the abundant words. If the length is short, we extend the review with 0s. After word embedding, each word

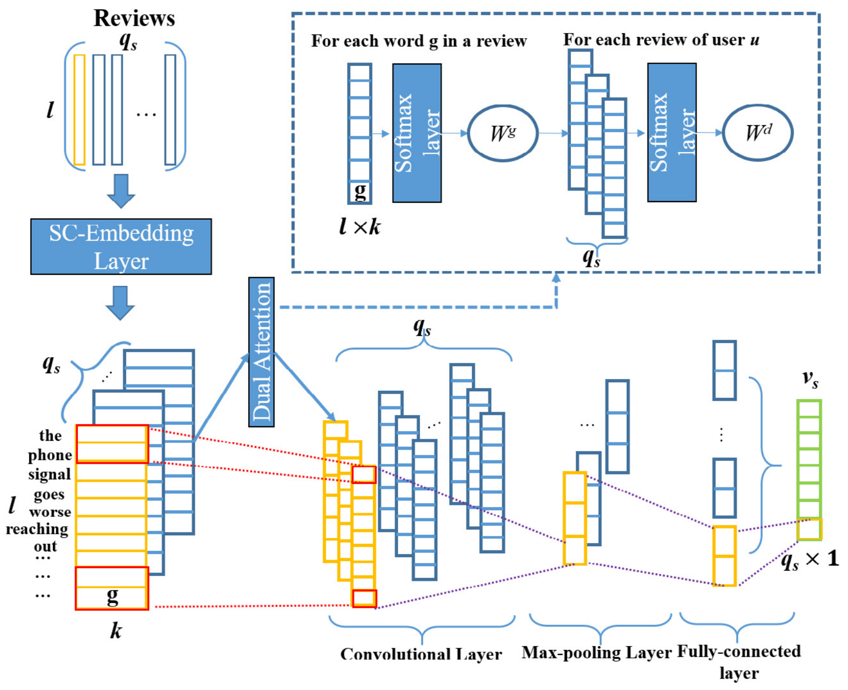


Fig. 3. Sentiment classification scoring module.

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

$$D_u = d_1 \oplus d_2 \oplus d_3 \cdots \oplus d_{q_s}, \tag{3}$$

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 of a review and each review of the review description matrix, both of which are named dual Attention Factors. It is obvious that different word plays a different role in a review, and different reviews of user u also affect the importance in different level (in our proposed model, the topic is the sentiment score we want to calculate). So we apply attention vector in words level. Meanwhile, as different reviews play different roles among all reviews, we apply attention vector in reviews level. This is why we apply dual-attention to calculate different weights for words and reviews and co-train the dual attention vectors with neural networks. Along this line, by the inspiration of Seo et al. (2017), we introduce attention vectors which are usually employed to profile latent descriptions for time sequences. Attention vector is a useful method to model the different importance of each instance in networks. With a combination, we call our attention vector "dual-attention vector". In our model, we first add one attention factor W^g for each word embedding g in each review of user u:

$$W^g d_i = w^g_1 g_1 \oplus w^g_2 g_2 \oplus w^g_3 g_3 \cdots \oplus w^d_l g_l,$$
$$W^g = \{w^g | \sum_{i=1}^{l} w^g_i = 1\}, \tag{4}$$

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

$$W^d D_u = w^d_1 d_1 \oplus w^d_2 d_2 \oplus w^d_3 d_3 \cdots \oplus w^d_{q_s} d_{q_s},$$
$$W^d = \{w^d | \sum_{i=1}^{q_s} w^d_i = 1\}, \tag{5}$$

where g stands for the k-dimensional embedding for each word. W^g and W^d stand for the attention vector for each word in review and each review in review matrix. As shown in Fig. 3, we utilize a softmax layer to gain weights of dual attention vectors.

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 also use one filter function to share 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 follows:

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

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

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

where W^4 and 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 described as follows:

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

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

We use users' ratings to train this convolutional neural network. Different from traditional methods, we do not transfer the ratings in training data (0–5) into six ranges in 0–1 for bi-sentiment analysis. Instead, we design the output of SC into six classes, which can be treated as a regularization in the output layer. Our model naturally matches the ratings of users (from 0 to 5) with a sentiment score. Finally , we compute the sentiment score vector v_s, whose entry is sentiment score s_{ui} with the range (0–5).

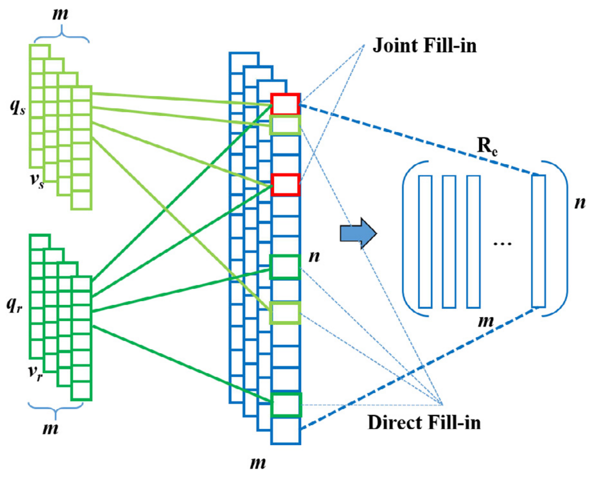


Fig. 4. Combination function module.

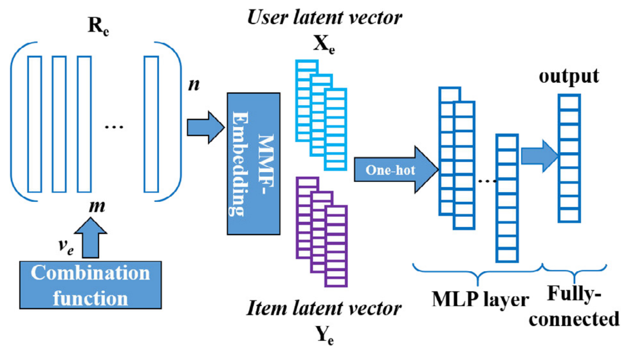


Fig. 5. MLP-based matrix factorization module.

5.2. Combination function

In this section, we introduce combination function, which can catch the opinion bias from reviews to ratings and enhance the original rating matrix. The process of combination function is shown as Fig. 4.

In combination function, we utilize the sentiment vector v_s for each user from sentiment scoring module. 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 yet 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-dimensional vector, v_s is a q_s-dimensional vector whose entry is s_{ui}, and v_r is a q_r-dimensional vector whose entry is r_{ui}, where q_r equals 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 alone, some important information on the other side will not be accessible, so it is difficult for us to tackle unbalanced data. Meanwhile, some users may make some useless reviews for some reasons (malicious reviews, rush comments, 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 s_{ui} and r_{ui} 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 the following three situations:

- **Direct Fill-in**: If the user u only rates or reviews the item i, $\otimes$ will fill v_e with either r_{ui} or s_{ui}.
- **Joint Fill-in**: If the user u rates and reviews the item i, and does not be dropped, $\otimes$ will fill 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 is apparent 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. Specifically, drop operation in Combination Function can be a potential tool for abnormal user detection. Moreover, we will explain how we decide weight ε in **Joint Fill-in**. Also μ, ι will be discussed in the experiment section. Finally, with the combination function, we arrange the joint vector v_e directly to build an enhanced rating matrix R_e. The process of Combination Function can be described as Eq. (11), where β denotes $(\varepsilon, \mu, \iota)$:

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

5.3. MLP-based matrix factorization

After the combination function, we get an enhanced rating matrix R_e. Then we build an MLP-based matrix factorization (MMF) to make recommendations with R_e. MMF module is composed of an embedding layer and MLP neural network. The reason why we utilize MLP as the recommendation model is:

(1) MLP is neural network based model, which has its strong ability to cope with non-linear situations in recommender system (He, Zhang et al., 2016).

(2) Comparing with some state-of-the-art NN-based model, MLP is relatively simple but effective in our proposed problem, as introduced in Section 2. Moreover, MLP can be easily extended to a more accurate and complex NN-based model.

(3) With the consideration of effectiveness and accuracy, we believe MLP is the proper NN framework for our proposed model and application scenarios. Fig. 5 shows the process of how we build the MMF module.

Note that R_e has the same size as original matrix R. Therefore, 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, $\tau = \{\tau_1, \tau_2, \ldots, \tau_p\}$, users and items can be mapped into latent vectors X_e and Y_e:

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

where Σ_e is a $p \times p$ 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:

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

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 both 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 (He et al., 2017). 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 tth 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 the activation function:

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

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

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

where T is the number of hidden layers 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:

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

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.

5.4. Model solution

When we train our proposed model, we employ a simple square function for both SC and MMF:

$$L^{SC} = \sum_{u \in U} \sum_{i \in (I \subset V)} w^{SC}(r_{ui} - s_{ui})^2, \tag{17}$$

$$L^{MMF} = \sum_{u \in U} \sum_{i \in I} w^{MMF}(r_{ui} - \tilde{r}_{ui})^2, \tag{18}$$

where w^{SC} and w^{MMF} are employed to avoid over-fitting. In specific, we utilize Stochastic Gradient Descent (SGD) to solve our model. The details of the whole process of NeuO is shown in Algorithm 1.

Algorithm 1 NeuO

```
Input: Training data: User-item matrix $R$, Review $V$, hyper-
    parameter $O$,
    Test data: User-item matrix $R_{test}$, Review $V_{test}$
Output: $\alpha, \beta, \gamma, \tau$, recommendation list $R-list$
 1: Initialization with $O$
 2: Train:
 3: Input $V$ to SC
 4: repeat
 5:     Compute Gradient of Eq. (17) using SGD
 6:     Update $\alpha$
 7: until convergence
 8: Define $\beta$
 9: Input $v_s$ $v_r$ to Combination Function
10: Input $R_e$ to MMF
11: repeat
12:     Compute Gradient of Eq. (18) using SGD
13:     Update $\gamma, \tau$
14: until convergence
15: Save the structure of NeuO $(\alpha, \beta, \gamma, \tau)$
16: Recommendation:
17: Build NeuO with $\alpha, \beta, \gamma, \tau, O$
18: Compute $v_s$ with Eq. (9)
19: Compute $R_e$ with Eq. (11)
20: Compute $\tilde{r}$ with Eq. (16)
21: $R-list$=Top-k($\tilde{r}$) for each user $u$
22: return $\alpha, \beta, \gamma, \tau$, recommendation list $R-list$
```

6. Experiments

6.1. Dataset

In order to validate the effectiveness of our model, we conduct extensive experiments on datasets from Amazon.com[2] and Yelp for RecSys.[3] Amazon and Yelp datasets are well-known public recommendation datasets with textual information. But both datasets are pre-filtered, so that they cannot validate the robustness for unbalance and opinion bias. Hence, we self-collect two real-world datasets from Taobao[4] and Jingdong[5] as supplementary to validate our method. All the datasets contain rating range from 0 to 5, and we use 5 fold 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 2. From Table 2, we can see that these datasets are extremely sparse, especially for the Taobao dataset with 49,053 user rating on items but only 10,791 reviews, which indicating it is a highly unbalanced dataset. So as to Jingdong dataset.

Table 2
The datasets' characteristics.

Dataset	Amazon	Yelp	Taobao	Jingdong
#user	30,759	45,980	10,121	8031
#item	16,515	11,537	9892	3025
#review	285,644	229,900	10,791	8310
#rating	285,644	229,900	49,053	25,152
Sparsity	0.051%	0.043%	0.049%	0.12%
Avg words /s	10.1	9.9	12.7	13.2
Avg words /r	104	130	65	70
Avg sentences /r	9.7	11.9	4.9	5.1
Avg reviews /u	9.29	5.00	1.06	1.03

6.2. Baselines

We compare our model with the following baselines:

(1) Attentive Collaborative Filtering (A-CF) (Chen et al., 2017): A-CF utilizes item- and component-level attention models to assign attentive weights for inferring the underlying users' preferences encoded in the implicit user feedback. And A-CF can achieve a superior performance over traditional collaborative filtering methods.

(2) Adaptive Matrix Factorization (A-MF) (Ning et al., 2017): A-MF is designed to learn personal models based on adapting the popular gradient descent optimization techniques. And A-MF can achieve a superior performance over traditional matrix factorization methods.

(3) Text-driven Latent Factor Model (TLFM) (Song, Gao, Feng, Wang, Wong, & Zhang, 2017): TLFM captures the semantics of reviews, user preferences and product characteristics by jointly optimizing two components, a user-specific LFM and a product-specific LFM. TLFM achieves state-of-the-art performance as other LDA-based methods.

[2] https://jmcauley.ucsd.edu/data/amazon.
[3] https://www.kaggle.com/c/yelp-recsys-2013.
[4] https://www.taobao.com.
[5] https://re.jd.com/.

Table 3
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**
Jingdong	1.672	1.555	1.5104	1.332	**1.250**

Table 4
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**
Jingdong	0.087	0.092	0.055	0.110	**0.166**

(4) Convolutional matrix factorization (ConvMF+) (Kim, Park, Oh, Lee, & Yu, 2016): ConvMF+ is a recently proposed context-aware recommendation model, based on the GloVe embedding to represent reviews and make recommendations. ConvMF+ is a CNN-based recommendation method.

6.3. *Parameter settings*

We initialize some parameters: Word Embedding Dimension $L = 130$, learning rate $= 0.0001$ with optimizer Adam, different length of filters in SC (2, 3, 4), joint weight in Combination Function $\varepsilon = 0.5$, $\mu = 3$, $\iota = 0.85$. And we also define the factors of full-connected layers in SC as 50. 0.45 dropout probability is used for the full-connected layers to reduce over-fitting. Activation functions of convolutional layers in both SC and MMF are ReLUs, and Softmax for output layer and dual attention vectors calculations. We also set the parameters of baselines as (Chen et al., 2017; Kim et al., 2016; Ning et al., 2017; Song et al., 2017) to make a plain comparison. Our experiments are operated with Pytorch running on GPU: GeForce GTX TITAN X.

6.4. *Experimental results and discussions*

6.4.1. *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 3, and HR results are shown in Table 4.

From MSE and HR results, we can see NeuO can achieve a superior performance than state-of-the-art baselines on all datasets, which clearly proves the effectiveness of our model. We further discuss the details of the experimental results. A-CF and A-MF can achieve good recommendations in a dense dataset. While in this comparison, they perform worse than other baselines with the limit of data sparsity and unbalance. Moreover, TLFM is a recent novel LDA-based method which utilizes reviews to make recommendations. And the ConvMF+ model integrates two subparts: (1) Probabilistic Matrix Factorization (PMF) for combining latent user and item vectors and (2) CNN for item contextual modeling. Note that TLFM and ConvMF+ can achieve the same level performance as NeuO on Amazon and Yelp. However, when applied to unbalanced Taobao and Jingdong dataset, our proposed model overperforms all the baselines with a large gap (especially on HR). The possible reason is that NeuO is designed with the consideration of unbalanced data in real-world applications. And the Combination Function in NeuO can utilize both reviews and ratings, which enhance the expression ability of original datasets. The experiment shows that our method can be applied in real-world unbalanced dataset and achieves a superior performance than baselines.

6.4.2. *Robustness for unbalanced data*

In order to verify the robustness of NeuO, we need to use some deliberately constructed datasets based on Amazon and Yelp (Taobao dataset and Jingdong dataset are already unbalanced) as real-world scenarios. By removing reviews from both rated and reviewed items randomly, we build two new datasets Amazon-unb and Yelp-unb (drop 0% to 100% reviews with 10% per group). When we drop 100% reviews, both datasets become rating-only datasets. The results of all baselines are shown in Fig. 6 as well as our model.

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 Tables 3 and 4), when the datasets become 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 achieves a stable performance like A-CF and A-MF. This experiment shows the ability of our proposed method to tackle unbalanced datasets and achieve stable performance.

6.4.3. *Parameter decision*

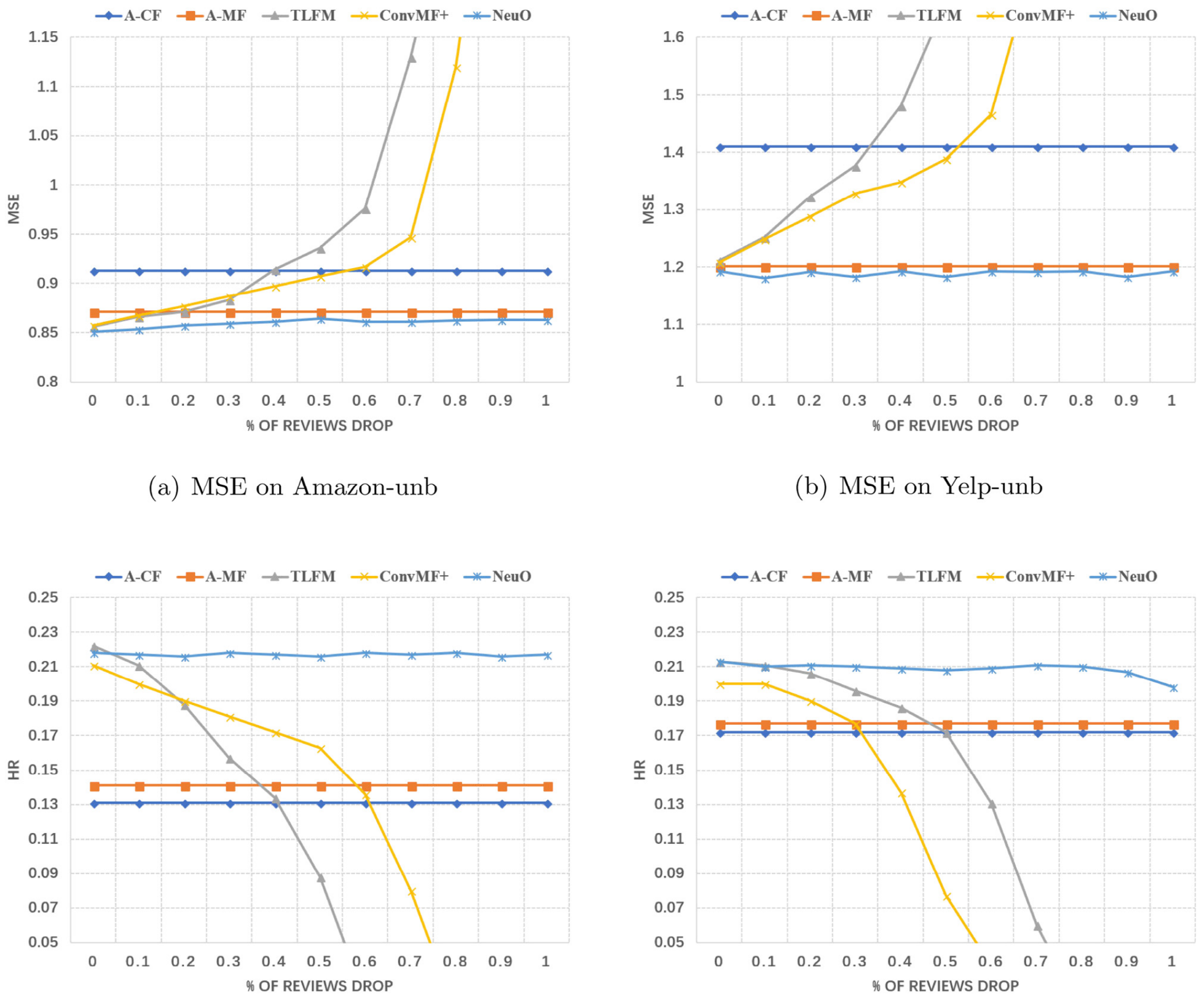
In order to achieve the best performance of our model, we need to predefine several parameters. Among all these parameters, we believe that the parameters in Combination Function, ε, μ, ι are very important to our model. ε is the combination weight between rating vector and review vector, μ is the threshold of opinion bias for dropping reviews and ι is the threshold for dropping users.

We conduct some experiments to decide ε, μ, ι on Amazon-unb and Taobao. The results are shown in Fig. 7. From Figs. 7(a) and 7(d), we find that combination weight ε affects our proposed method on real-world dataset (Taobao) rather than our hand-made dataset (Amazon-unb). When ε is small, rating vectors play an important role in NeuO and NeuO can be a rating-based model like A-MF and A-CF. While ε is large, NeuO becomes a review-based model like ConvMF+. So our model is a general framework which is suitable for different data situations. When we choose the proper ε, we could achieve the best performance of NeuO (0.521 is the best weight). Without loss of generality, we set $\varepsilon = 0.5$ as our default settings.

From Figs. 7(b), 7(c), 7(e) and 7(f), it indicates that μ, ι do not make sense in Amazon-unb because the partition of opinion bias in this pre-filtered dataset is so small. However, when applied to real-world dataset Taobao, it is noticeable that our method with Combination Function improves the performance a lot. The opinion bias in real-world is a quite often phenomenon. When the thresholds μ, ι are small, combination function loses the ability to filter the opinion bias and fades to a simple combination of sentiment and ratings. According to our experiment, we choose $\mu = 3$, $\iota = 0.85$ as our default settings.

6.4.4. *Effect of opinion bias*

One advantage of our proposed model NeuO is its ability to catch the bias from reviews to ratings, and build an enhanced matrix R_e to utilize both. To validate the effect of opinion bias, we conduct experiments on all four datasets: A-unb(Amazon-unb), Y-unb(Yelp-unb) (both 20% dropped), Taobao and JD(Jingdong). Especially on real-world dataset Taobao and Jingdong, they show exactly what we can do with the opinion bias.



(a) MSE on Amazon-unb

(b) MSE on Yelp-unb

(c) HR on Amazon-unb

(d) HR on Yelp-unb

Fig. 6. Robustness for unbalanced dataset with different models.

Table 5
Effect of opinion bias.

	A-unb		Y-unb		Taobao		JD	
MSE	R	R_e	R	R_e	R	R_e	R	R_e
A-CF	0.973	**0.942**	1.41	**1.31**	1.512	**1.342**	1.444	**1.371**
A-MF	0.873	**0.852**	1.201	**1.194**	1.341	**1.197**	1.603	**1.412**
MMF	0.875	**0.851**	1.204	**1.193**	1.356	**1.195**	1.333	**1.200**
HR								
A-CF	0.131	**0.170**	0.172	**0.200**	0.091	**0.161**	0.088	**0.111**
A-MF	0.141	**0.176**	0.177	**0.205**	0.089	**0.156**	0.078	**0.121**
MMF	0.150	**0.218**	0.174	**0.213**	0.094	**0.185**	0.097	**0.124**

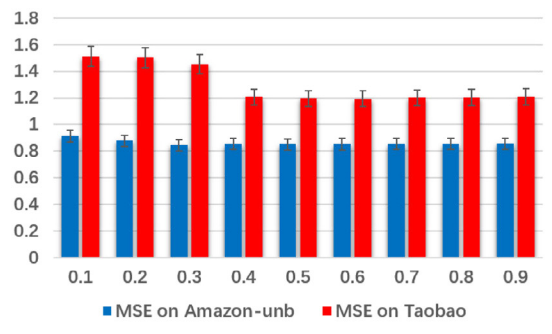
We feed enhanced matrix R_e and original R separately into our method NeuO and two rating-based baselines: A-MF and A-CF to see the effect of opinion bias of NeuO.

The results in Table 5 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 for this are (1) Combination Function utilizes both ratings and reviews, which enrich 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).

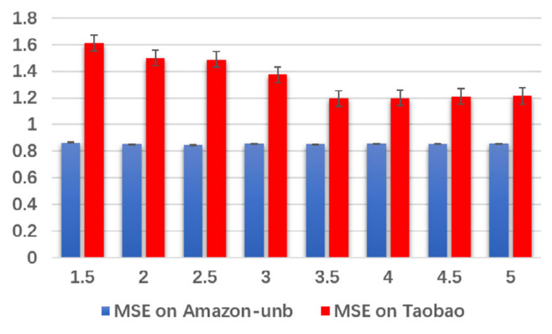
Moreover, we show a potential application of opinion bias on one real-world dataset Taobao. There are always complex scenarios in a real e-commerce website: some malicious users try to make profits from websites by giving a negative review on purpose. They may write some non-relative comments but give 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 in Taobao, NeuO dropped 7 users and 142 reviews. Some desensitization information of these users are shown in Table 6.

All those users rate a lot negative reviews and ratings among all the reviews and ratings (except *user*7, which is another type of malicious user). Moreover, we use *user*5 and *user*7 as examples, by plotting the sentiment vector v_s and rating vector v_r.

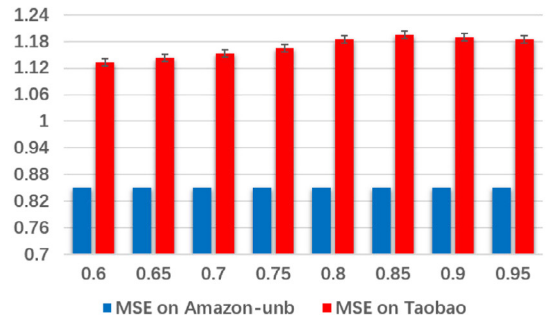
From Fig. 8, we can see that *user*5 is the one who always gives a good review but a low rating (almost gets sentiment score 5 for all 1-star ratings) and *user*7 is the one who always gives a negative review but a high positive rating. According to the opinion bias caught by NeuO, we can confirm that both two users are abnormal



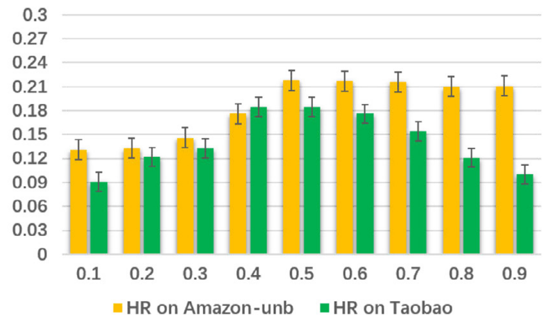
(a) Combination weight ε



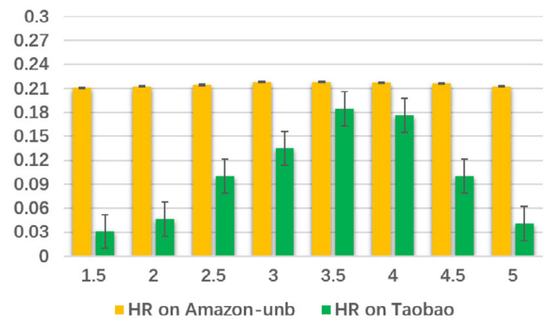
(b) Opinion bias threshold μ



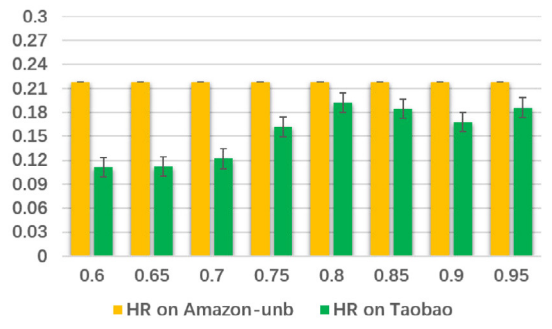
(c) Drop user threshold ι



(d) Combination weight ε

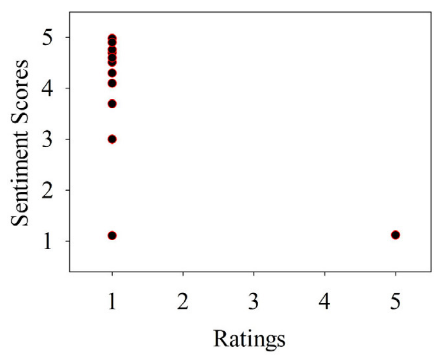


(e) Opinion bias threshold μ

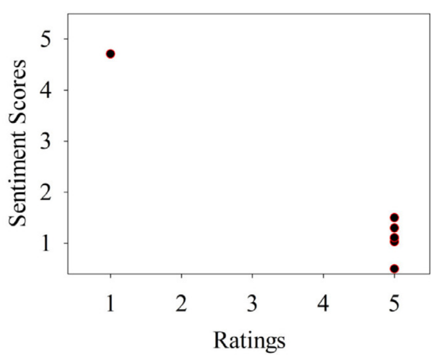


(f) Drop user threshold ι

Fig. 7. Parameter decision for NeuO.



(a) Ratings and senti-score of *user*5



(b) Ratings and senti-score of *user*7

Fig. 8. Reason for malicious user detection.

for e-commerce websites and could be malicious users. It shows the potential of NeuO to be applied to some special applications in real scenarios.

6.4.5. *Effect of dual attention vectors*

The idea of Attention vectors is often applied to Sequence learning (Sutskever, Vinyals, & Le, 2014) in speech translation domain.

Table 6
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

Table 7
Effect of dual attention vector.

	Amazon	Yelp	Taobao	Jingdong
MSE				
No-at	1.112(+0%)	1.302(+0%)	1.333(+0%)	1.412(+0%)
W^d	0.937(+6.3%)	1.199(+7.9%)	1.221(+8.4%)	1.313(+7.0%)
W^g	0.912(+8.8%)	1.232(+5.3%)	1.210(+9.2%)	1.400(+0.8%)
W^d, W^g	**0.851(+14.9%)**	**1.193(+8.3%)**	**1.195(+10.3%)**	**1.250(+11.5%)**
HR				
No-at	0.146(+0%)	0.182(+0%)	0.168(+0%)	0.142(+0%)
W^d	0.156(+6.8%)	0.183(+0.54%)	0.181(+7.7%)	0.151(+6.3%)
W^g	0.175(+19.8%)	0.198(+8.7%)	0.182(+8.3%)	0.161(+13.3%)
W^d, W^g	**0.218(+49%)**	**0.213(+17%)**	**0.185(+10.11%)**	**0.166(+16.9%)**

Attention method can be treated as a self-adaptive weights decision method, which means that it can compute different weights for different factor in a sequence. In NeuO, we need to compute the sentiment score according to the reviews of users. So we treat one review as a sequence of words and one user's review matrix as a sequence of reviews. Then we borrow the idea of attention vectors and apply dual attention vectors to words in review and reviews in review matrix both: W^g and W^d. W^g is utilized for ensuring the different weights for different words in a review while W^d is utilized ensuring the different weights for different reviews in all the reviews of a special user. The effect of dual attention vectors is shown as follows:

From the results in Table 7, it indicates that (1) applying W^d and W^g separately can improve the performance of NeuO, which means both attention vectors make sense in our recommender system. (2) Applying dual attention vectors (W^d+W^g) can improve our proposed method NeuO significantly on different datasets (Amazon 20%, Yelp 10%, Taobao 10% and Jingdong 10%), which ensures the effects of dual attention vectors in our proposed model.

7. Conclusion

In this paper, we proposed a simple, extensible two-step training neural network recommender system, NeuO, which focused on capturing opinion bias between review content and users' ratings in unbalanced datasets. We also proposed SC, a CNN with dual attention vectors to predict the sentiment scores for users' reviews. A well-designed combination function was employed to catch opinion bias utilizing both ratings and review scores and built an enhanced user–item matrix. Finally, we employed MMF to make the recommendations. The experimental results on Amazon, Yelp and Taobao datasets showed that (1) our method outperforms state-of-the-art baselines in accuracy. (2) NeuO achieves a stable performance in unbalanced datasets. (3) NeuO is able to catch the opinion bias correctly and has the potential to be applied to real-world applications.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant No. 61772230, 61773361, U1836206 and Natural Science Foundation of China for Young Scholars No. 61702215, China Postdoctoral Science Foundation No. 2017M611322 and 2018T110247.

References

Adomavicius, G., & Tuzhilin, A. (2015). Context-aware recommender systems. In *Recommender systems handbook* (pp. 191–226). Springer.

Bai, T., Wen, J. -R., Zhang, J., & Zhao, W. X. (2017). A neural collaborative filtering model with interaction-based neighborhood. In *Proceedings of the 2017 ACM on conference on information and knowledge management* (pp. 1979–1982). ACM.

Beel, J., Gipp, B., Langer, S., & Breitinger, C. (2016). paper recommender systems: a literature survey. *International Journal on Digital Libraries*, *17*(4), 305–338.

Champiri, Z. D., Shahamiri, S. R., & Salim, S. S. B. (2015). A systematic review of scholar context-aware recommender systems. *Expert Systems with Applications*, *42*(3), 1743–1758.

Chen, J., Zhang, H., He, X., Nie, L., Liu, W., & Chua, T. -S. (2017). 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* (pp. 335–344). ACM.

Chin, W. -S., Zhuang, Y., Juan, Y. -C., & Lin, C. -J. (2015). A fast parallel stochastic gradient method for matrix factorization in shared memory systems. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *6*(1), 2.

Han, J., Zuo, W., Liu, L., Xu, Y., & Peng, T. (2016). Building text classifiers using positive, unlabeled and 'outdated' examples. *Concurrency Computations: Practice and Experience*, *28*(13), 3691–3706.

Harper, F., & Konstan, J. A. (2016). The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems*, *5*(4), 19.

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. -S. (2017). Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web* (pp. 173–182). International World Wide Web Conferences Steering Committee.

He, C., Parra, D., & Verbert, K. (2016). Interactive recommender systems: A survey of the state of the art and future research challenges and opportunities. *Expert Systems with Applications*, *56*, 9–27.

He, X., Zhang, H., Kan, M. -Y., & Chua, T. -S. (2016). Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 39th international ACM SIGIR conference on research and development in information retrieval* (pp. 549–558). ACM.

Kim, D., Park, C., Oh, J., Lee, S., & Yu, H. (2016). Convolutional matrix factorization for document context-aware recommendation. In *Proceedings of the 10th ACM conference on recommender systems* (pp. 233–240). ACM.

Koren, Y., & Bell, R. (2015). Advances in collaborative filtering. In *Recommender systems handbook* (pp. 77–118). Springer.

Lian, J., Zhang, F., Xie, X., & Sun, G. (2017). CCCFNet: a content-boosted collaborative filtering neural network for cross domain recommender systems. In *Proceedings of the 26th international conference on world wide web companion* (pp. 817–818). International World Wide Web Conferences Steering Committee.

Liu, Q., Wu, S., & Wang, L. (2015). COT: Contextual operating tensor for context-aware recommender systems.,

Lu, J., Wu, D., Mao, M., Wang, W., & Zhang, G. (2015). Recommender system application developments: a survey. *Decision Support Systems*, *74*, 12–32.

Luca, M., & Zervas, G. (2016). Fake it till you make it: Reputation, competition, and Yelp review fraud. *Management Science*, *62*(12), 3412–3427.

Ning, Y., Shi, Y., Hong, L., Rangwala, H., & Ramakrishnan, N. (2017). A gradient-based adaptive learning framework for efficient personal recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems* (pp. 23–31). ACM.

Panniello, U., Tuzhilin, A., & Gorgoglione, M. (2014). Comparing context-aware recommender systems in terms of accuracy and diversity. *User Modeling and User-Adapted Interaction*, *24*(1–2), 35–65.

Rosenthal, S., Farra, N., & Nakov, P. (2017). SemEval-2017 task 4: Sentiment analysis in Twitter. In *Proceedings of the 11th international workshop on semantic evaluation* (pp. 502–518).

Rubens, N., Elahi, M., Sugiyama, M., & Kaplan, D. (2015). Active learning in recommender systems. In *Recommender systems handbook* (pp. 809–846). Springer.

Seo, S., Huang, J., Yang, H., & Liu, Y. (2017). Interpretable convolutional neural networks with dual local and global attention for review rating prediction. In *Proceedings of the eleventh ACM conference on recommender systems* (pp. 297–305). ACM.

Song, K., Gao, W., Feng, S., Wang, D., Wong, K., & Zhang, C. (2017). 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* (pp. 2744–2750). AAAI Press.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems* (pp. 3104–3112).

Wei, J., He, J., Chen, K., Zhou, Y., & Tang, Z. (2017). Collaborative filtering and deep learning based recommendation system for cold start items. *Expert Systems with Applications*, *69*, 29–39.

Xue, B., Fu, C., & Shaobin, Z. (2014). A study on sentiment computing and classification of sina weibo with word2vec. In *Big data (BigData Congress), 2014 IEEE international congress on* (pp. 358–363). IEEE.

Yang, C., Bai, L., Zhang, C., Yuan, Q., & Han, J. (2017). Bridging collaborative filtering and semi-supervised learning: a neural approach for poi recommendation. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1245–1254). ACM.

Yang, X., Guo, Y., Liu, Y., & Steck, H. (2014). A survey of collaborative filtering based social recommender systems. *Computer Communications*, *41*, 1–10.

Yang, B., Lei, Y., Liu, J., & Li, W. (2017). Social collaborative filtering by trust. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(8), 1633–1647.

Yang, J., Liu, C., Teng, M., Chen, J., & Xiong, H. (2017). A unified view of social and temporal modeling for B2B marketing campaign recommendation. *IEEE Transactions on Knowledge and Data Engineering*.

Yang, Y., Xu, Y., Wang, E., Han, J., & Yu, Z. (2017). Improving existing collaborative filtering recommendations via serendipity-based algorithm. *IEEE Transactions on Multimedia*.

Ying, H., Zhuang, F., Zhang, F., Liu, Y., Xu, G., Xie, X., Xiong, H., & Wu, J. (2018). Sequential recommender system based on hierarchical attention networks. In *Proceedings of the 27th international joint conference on artificial intelligence*.

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

Zhuang, F., Luo, D., Yuan, N. J., Xie, X., & He, Q. (2017). Representation learning with pair-wise constraints for collaborative ranking. In *Proceedings of the tenth ACM international conference on web search and data mining* (pp. 567–575). New York, NY, USA: ACM.

Zhuang, F., Luo, D., Yuan, N. J., Xie, X., & He, Q. (2017a). Representation learning with pair-wise constraints for collaborative ranking. In *Proceedings of the tenth ACM international conference on web search and data mining* (pp. 567–575). ACM.

Zhuang, F., Zhang, Z., Qian, M., Shi, C., Xie, X., & He, Q. (2017). Representation learning via Dual-Autoencoder for recommendation. *Neural Networks*, *90*, 83–89.