Order-Aware Embedding Non-sampling Factorization Machines for Context-Aware Recommendation

Qingzhi Hou[1], Yifeng Chen[2,3,4], Mei Yu[2,3,4], Ruiguo Yu[2,3,4], Jian Yu[2,3,4], Mankun Zhao[2,3,4], Tianyi Xu[2,3,4], and Xuewei Li[2,3,4](✉)

[1] School of Civil Engineering, Tianjin University, Tianjin, China
qhou@tju.edu.cn
[2] College of Intelligence and Computing, Tianjin University, Tianjin, China
{haichuang,yumei,rgyu,yujian,zmk,tianyi.xu,lixuewei}@tju.edu.cn
[3] Tianjin Key Laboratory of Cognitive Computing and Application, Tianjin, China
[4] Tianjin Key Laboratory of Advanced Networking (TANK Lab), Tianjin, China

Abstract. FM can use the second-order feature interactions. Some researchers combine FM with deep learning to get the high-order interactions. However, these models rely on negative sampling. ENSFM adopts non-sampling and gets fine results, but it does not consider the high-order interactions. In this paper, we add the high-order interactions to ENSFM. We also introduce a technique called Order-aware Embedding. The excellent results show the effectiveness of our model.

Keywords: Context-aware recommendation · Factorization machines · Non-sampling · The high-order interactions · Order-aware embedding

1 Introduction

ENSFM [1] achieves non-sampling, but only considers the second-order interactions. In this paper, we continue to use non-sampling. On this basis, the third-order and the fourth-order interactions are added. We also consider that the use of shared embedding may cause some problems. Therefore, we adopt a technique called Order-aware Embedding to solve these problems. Its main idea is to apply different embeddings to different orders for feature interactions.

The main contributions of this work are summarized as follows: (1)We consider that the high-order interactions have an important influence on performance, so the third-order and the fourth-order interactions are added. (2)We believe that the use of shared embedding will result in learned feature interactions less effective, so we adopt Order-aware Embedding.

2 Preliminaries

2.1 Factorization Machines (FM)

FM is a machine learning algorithm based on MF. The model uses a low-dimensional dense vector to represent the weight of a feature. The number of

H. Yang et al. (Eds.): ICONIP 2020, CCIS 1332, pp. 780–788, 2020.
https://doi.org/10.1007/978-3-030-63820-7_89

user features and item features are denoted by m and n, respectively. By using the factorized parameters, FM captures all interactions between features:

$$\hat{y}_{FM}(x) = w_0 + \sum_{i=1}^{m+n} w_i x_i + \sum_{i=1}^{m+n} \sum_{j=i+1}^{m+n} e_i^T e_j \cdot x_i x_j \tag{1}$$

2.2 Efficient Non-sampling Matrix Factorization

Although the performance of non-sampling matrix factorization is excellent, its shortcoming is also obvious—inefficiency. In order to solve this problem, researchers have proposed some effective solutions [2,11,12].

Theorem 1. *A generalized matrix factorization whose prediction function is:*

$$\hat{y}_{uv} = \mathbf{h}^T \left(\mathbf{p}_u \odot \mathbf{q}_v \right) \tag{2}$$

where $\mathbf{p}_u$ *and* $\mathbf{q}_v$ *are representation vectors of user and item, respectively. And* $\odot$ *denotes the element-wise product of two vectors. Its loss function is:*

$$\mathcal{L}(\Theta) = \sum_{u \in \mathbf{U}} \sum_{v \in \mathbf{V}} c_{uv} \left(y_{uv} - \hat{y}_{uv} \right)^2 \tag{3}$$

where c_{uv} *is the weight of sample* y_{uv}*. It is completely equivalent to that of:*

$$\begin{aligned} \tilde{\mathcal{L}}(\Theta) = & \sum_{u \in \mathrm{U}} \sum_{v \in \mathrm{V}^+} \left(\left(c_{uv}^+ - c_{uv}^- \right) \hat{y}_{uv}^2 - 2 c_{uv}^+ \hat{y}_{uv} \right) \\ & + \sum_{i=1}^{d} \sum_{j=1}^{d} \left(\left(h_i h_j \right) \left(\sum_{u \in \mathrm{U}} p_{u,i} p_{u,j} \right) \left(\sum_{v \in \mathrm{V}} c_{uv}^- q_{v,i} q_{v,j} \right) \right) \end{aligned} \tag{4}$$

3 Our Model—ONFM

3.1 Overview

Using the FM form, ONFM is expressed as:

$$\hat{y}_{FM}(x) = w_0 + \sum_{i=1}^{m+n} w_i x_i + f_2(x) + f_3(x) + f_4(x) \tag{5}$$

$$f_2(x) = h_r \sum_{i=1}^{m+n} \sum_{j=i+1}^{m+n} \left(x_i e_i^2 \odot x_j e_j^2 \right) \tag{6}$$

$$f_3(x) = h_s \sum_{i=1}^{m+n} \sum_{j=i+1}^{m+n} \sum_{k=j+1}^{m+n} \left(x_i e_i^3 \odot x_j e_j^3 \odot x_k e_k^3 \right) \tag{7}$$

$$f_4(x) = h_t \sum_{i=1}^{m+n} \sum_{j=i+1}^{m+n} \sum_{k=j+1}^{m+n} \sum_{l=k+1}^{m+n} \left(x_i e_i^4 \odot x_j e_j^4 \odot x_k e_k^4 \odot x_l e_l^4 \right) \tag{8}$$

where $f_2(x)$, $f_3(x)$ and $f_4(x)$ denote the second-order, the third-order and the fourth-order, respectively. Figure 1 shows the composition structure of ONFM.

There are five layers—Input, Order-aware Embedding, Feature Pooling, Fully-connected and Output. The input of Input are some high-dimensional sparse vectors obtained by one-hot encoding. We need to convert these special vectors into low-dimensional dense vectors. The role of Order-aware Embedding is to solve this problem. After Order-aware Embedding processing, we get three different sets of low-dimensional dense vectors. The embedding vectors of feature i for different orders can be formulated as [5]:

$$e_i^j = W_i^j X \left[start_i : end_i \right] \tag{9}$$

Then these low-dimensional dense vectors directly enter Feature Pooling for feature interaction processing. The target of this layer is to reconstruct FM model in Eq. (5) into a completely equivalent generalized MF form:

$$\hat{y}_{FM}(\mathbf{x}) = \mathbf{h}_{aux}^T \left(\mathbf{p}_u \odot \mathbf{q}_v \right) \tag{10}$$

where $\mathbf{p}_u$, $\mathbf{q}_v$ are two vectors obtained by Feature Pooling. They are only related to the corresponding user and item, not to the objects they interact with.

Finally, the two vectors are input to Fully-connected. Then, we obtain the final prediction $\hat{y}_{FM}$, which represents user u's preference for item v.

3.2 ONFM Theoretical Analysis

The basic theory of ONFM is that the FM model incorporating the high-order interactions in Eq. (5) can be transformed into a MF form in Eq. (10). Then we will prove the correctness of the theory.

Recalling Eq. (5), we consider three parts—$f_2(x)$, $f_3(x)$, $f_4(x)$, they can be transformed into the following form:

$$\begin{aligned} f_2(x) = {} & h_1 \left(\sum_{i=1}^{m} \sum_{j=i+1}^{m} \left(x_i^u e_i^{u,2} \odot x_j^u e_j^{u,2} \right) + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \left(x_i^v e_i^{v,2} \odot x_j^v e_j^{v,2} \right) \right) \\ & + h_2 \left(\sum_{i=1}^{m} x_i^u e_i^{u,2} \odot \sum_{i=1}^{n} x_i^v e_i^{v,2} \right) \end{aligned} \tag{11}$$

$$\begin{aligned} f_3(x) = {} & h_3(a+b) + h_4 \left(\sum_{i=1}^{m} \sum_{j=i+1}^{m} \left(x_i^u e_i^{u,3} \odot x_j^u e_j^{u,3} \right) \odot \sum_{i=1}^{n} x_i^v e_i^{v,3} \right) \\ & + h_4 \left(\sum_{i=1}^{m} x_i^u e_i^{u,3} \odot \sum_{i=1}^{n} \sum_{j=i+1}^{n} \left(x_i^v e_i^{v,3} \odot x_j^v e_j^{v,3} \right) \right) \end{aligned} \tag{12}$$

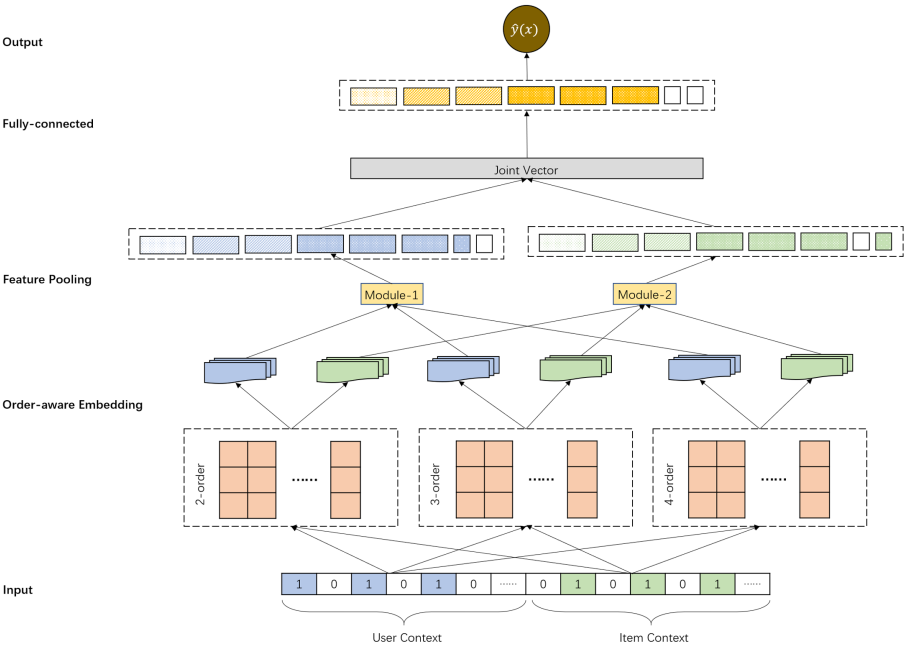

Fig. 1. The overall framework of ONFM.

$$a = \sum_{i=1}^{m} \sum_{j=i+1}^{m} \sum_{k=j+1}^{m} \left(x_i^u e_i^{u,3} \odot x_j^u e_j^{u,3} \odot x_k^u e_k^{u,3} \right) \tag{13}$$

$$b = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \sum_{k=j+1}^{n} \left(x_i^v e_i^{v,3} \odot x_j^v e_j^{v,3} \odot x_k^v e_k^{v,3} \right) \tag{14}$$

$$\begin{aligned} f_4(x) = {} & h_5(c+d) \\ & + h_6 \left(\sum_{i=1}^{m} \sum_{j=i+1}^{m} \sum_{k=j+1}^{m} \left(x_i^u e_i^{u,4} \odot x_j^u e_j^{u,4} \odot x_k^u e_k^{u,4} \right) \odot \sum_{i=1}^{n} x_i^v e_i^{v,4} \right) \\ & + h_6 \left(\sum_{i=1}^{m} x_i^u e_i^{u,4} \odot \sum_{i=1}^{n} \sum_{j=i+1}^{n} \sum_{k=j+1}^{n} \left(x_i^v e_i^{v,4} \odot x_j^v e_j^{v,4} \odot x_k^v e_k^{v,4} \right) \right) \\ & + h_7 \left(\sum_{i=1}^{m} \sum_{j=i+1}^{m} \left(x_i^u e_i^{u,4} \odot x_j^u e_j^{u,4} \right) \odot \sum_{i=1}^{n} \sum_{j=i+1}^{n} \left(x_i^v e_i^{v,4} \odot x_j^v e_j^{v,4} \right) \right) \end{aligned} \tag{15}$$

$$c = \sum_{i=1}^{m} \sum_{j=i+1}^{m} \sum_{k=j+1}^{m} \sum_{l=k+1}^{m} \left(x_i^u e_i^{u,4} \odot x_j^u e_j^{u,4} \odot x_k^u e_k^{u,4} \odot x_l^u e_l^{u,4} \right) \tag{16}$$

$$d = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \sum_{k=j+1}^{n} \sum_{l=k+1}^{n} \left(x_i^v e_i^{v,4} \odot x_j^v e_j^{v,4} \odot x_k^v e_k^{v,4} \odot x_l^v e_l^{v,4} \right) \tag{17}$$

Next, we will describe the training process of ONFM in detail by constructing three auxiliary vectors—$\mathbf{p}_u$, $\mathbf{q}_v$ and $\mathbf{h}_{aux}$:

$$p_u = \begin{pmatrix} p_{u,d}^{2,1} \\ p_{u,d}^{3,1} \\ p_{u,d}^{3,2} \\ p_{u,d}^{4,1} \\ p_{u,d}^{4,2} \\ p_{u,d}^{4,3} \\ p_{u,1} \\ 1 \end{pmatrix}; q_v = \begin{pmatrix} q_{v,d}^{2,1} \\ q_{v,d}^{3,2} \\ q_{v,d}^{3,1} \\ q_{v,d}^{4,3} \\ q_{v,d}^{4,2} \\ q_{v,d}^{4,1} \\ 1 \\ q_{v,1} \end{pmatrix}; h_{aux} = \begin{pmatrix} h_{aux,d}^{2} \\ h_{aux,d}^{4} \\ h_{aux,d}^{4} \\ h_{aux,d}^{6} \\ h_{aux,d}^{7} \\ h_{aux,d}^{6} \\ 1 \\ 1 \end{pmatrix} \tag{18}$$

For the auxiliary vector $\mathbf{p}_u$, it is calculated by module-1. The input of module-1 are multiple sets of user feature embedding vectors. The first six elements have a unified format—$p_{u,d}^{x,y}$. They are used for the user-item feature interactions. The last two elements are related to the global weight, the first-order features and the self feature interactions. The form of each part is expressed as follows.

$$p_{u,d}^{2,1} = \sum_{i=0}^{m} x_i^u e_i^{u,2} \tag{19}$$

$$p_{u,d}^{3,1} = \sum_{i=0}^{m} x_i^u e_i^{u,3} \tag{20}$$

$$p_{u,d}^{3,2} = \sum_{i=0}^{m} \sum_{j=i+1}^{m} \left(x_i^u e_i^{u,3} \odot x_j^u e_j^{u,3} \right) \tag{21}$$

$$p_{u,d}^{4,1} = \sum_{i=0}^{m} x_i^u e_i^{u,4} \tag{22}$$

$$p_{u,d}^{4,2} = \sum_{i=0}^{m} \sum_{j=i+1}^{m} \left(x_i^u e_i^{u,4} \odot x_j^u e_j^{u,4} \right) \tag{23}$$

$$p_{u,d}^{4,3} = \sum_{i=0}^{m} \sum_{j=i+1}^{m} \sum_{k=j+1}^{m} \left(x_i^u e_i^{u,4} \odot x_j^u e_j^{u,4} \odot x_k^u e_k^{u,4} \right) \tag{24}$$

$$\begin{aligned} p_{u,1} = w_0 &+ \sum_{i=0}^{m} w_i^u x_i^u + h_1 \sum_{i=0}^{m} \sum_{j=i}^{m} \left(x_i^u e_i^{u,2} \odot x_j^u e_j^{u,2} \right) \\ &+ h_3 \sum_{i=1}^{m} \sum_{j=i+1}^{m} \sum_{k=j+1}^{m} \left(x_i^u e_i^{u,3} \odot x_j^u e_j^{u,3} \odot x_k^u e_j^{u,3} \right) \\ &+ h_5 \sum_{i=1}^{m} \sum_{j=i+1}^{m} \sum_{k=j+1}^{m} \sum_{l=k+1}^{m} \left(x_i^u e_i^{u,4} \odot x_j^u e_j^{u,4} \odot x_k^u e_k^{u,4} \odot x_l^u e_l^{u,4} \right) \end{aligned} \tag{25}$$

The second auxiliary vector $\mathbf{q}_v$ is similar to the first one, but the element position is adjusted accordingly. For the third auxiliary vector h_{aux}, $h^2_{\text{aux},d} = h_2$, $h^4_{\text{aux},d} = h_4$, $h^6_{\text{aux},d} = h_6$, $h^7_{\text{aux},d} = h_7$.

3.3 Optimization Method

Now, we will introduce some optimization methods for ONFM. Firstly, the theory we have proved shows that FM can be expressed as two uncorrelated vectors—$\mathbf{p}_u$ and $\mathbf{q}_v$. By precomputing the vectors, we can greatly improve the training efficiency. Secondly, after transforming FM into generalized MF, the prediction function of ONFM satisfies the requirements of Theorem 1, so the loss function it proposes is available for ONFM:

$$\begin{aligned}\tilde{\mathcal{L}}(\Theta) = &\sum_{u\in\mathbf{B}}\sum_{v\in\mathbf{V}^+}\left(\left(c_v^+ - c_v^-\right)\hat{y}(\mathbf{x})^2 - 2c_v^+\hat{y}(\mathbf{x})\right) \\ &+ \sum_{i=1}^{d}\sum_{j=1}^{d}\left((h_{aux,i}h_{aux,j})\left(\sum_{u\in\mathbf{B}} p_{u,i}p_{u,j}\right)\left(\sum_{v\in\mathbf{V}} c_v^- q_{v,i}q_{v,j}\right)\right)\end{aligned} \tag{26}$$

where $\mathbf{B}$ indicates a batch of users, and $\mathbf{V}$ indicates all items.

4 Experiments

4.1 Experimental Settings

Datasets. The two publicly available datasets used are ***Frapple*** and ***Last.fm***. For ***Frapple***, the number of user, item, feature and instance are 957, 4082, 5382, 96203. For ***Last.fm***, the number are 1000, 20301, 37358 and 214574.

Baseline. We compare ONFM with the following baseline:

- **PopRank:** This model returns Top-k most popular items.
- **FM** [9]**:** The original factorization machines.
- **NFM** [6]**:** Neural factorization machine uses MLP to learn nonlinear and high-order interactions signals.
- **DeepFM** [4]**:** This model combines FM and MLP to make recommendations.
- **ONCF** [7]**:** This model improves MF with outer product.
- **CFM** [10]**:** Convolutional Factorization Machine uses 3D convolution to achieve the high-order interactions between features.
- **ENMF** [3]**:** Efficient Neural Matrix Factorization uses non-sampling neural recommendation method to generate recommendations.
- **ENSFM** [1]**:** Efficient Non-Sampling Factorization Machines conducts non-sampling training by transforming FM into MF.

Table 1. The performance of different models on Frappe and List.fm.

Frappe	HR@5	HR@10	HR@20	NDCG@5	NDCG@10	NDCG@20
PopRank	0.2539	0.3493	0.4136	0.1595	0.1898	0.2060
FM	0.4204	0.5486	0.6590	0.3054	0.3469	0.3750
DeepFM	0.4632	0.6035	0.7322	0.3308	0.3765	0.4092
NFM	0.4798	0.6197	0.7382	0.3469	0.3924	0.4225
ONCF	0.5359	0.6531	0.7691	0.3940	0.4320	0.4614
CFM	0.5462	0.6720	0.7774	0.4153	0.4560	0.4859
ENMF	0.5682	0.6833	0.7749	0.4314	0.4642	0.4914
ENSFM	0.6094	0.7118	0.7889	0.4771	0.5105	0.5301
ONFM-1	0.6149	0.7198	0.7927	0.4778	0.5119	0.5305
ONFM-2	0.6468	0.7623	0.8485	0.4978	0.5354	0.5574
ONFM-3	**0.6743**	**0.7924**	**0.8703**	**0.5217**	**0.5601**	**0.5800**
Last.fm	HR@5	HR@10	HR@20	NDCG@5	NDCG@10	NDCG@20
PopRank	0.0013	0.0023	0.0032	0.0007	0.0011	0.0013
FM	0.1658	0.2382	0.3537	0.1142	0.1374	0.1665
DeepFM	0.1773	0.2612	0.3799	0.1204	0.1473	0.1772
NFM	0.1827	0.2678	0.3783	0.1235	0.1488	0.1765
ONCF	0.2183	0.3208	0.4611	0.1493	0.1823	0.2176
CFM	0.2375	0.3538	0.4841	0.1573	0.1948	0.2277
ENMF	0.3188	0.4254	0.5279	0.2256	0.2531	0.2894
ENSFM	0.3683	0.4729	0.5793	0.2744	0.3082	0.3352
ONFM-1	0.4400	0.5386	0.6294	0.3306	0.3625	0.3856
ONFM-2	0.5431	0.6220	0.6822	0.4190	0.4446	0.4601
ONFM-3	**0.5673**	**0.6457**	**0.6946**	**0.4478**	**0.4733**	**0.4858**

Evaluation Protocols and Metrics. ONFM adopts the leave-one-out evaluation protocol [8,10] to test its performance. For Frappe, we randomly choice one transaction as the test example for each specific user context because of no timestamp. For List.fm, the latest transaction of each user is held out for testing and the rest is treated as the training set. The evaluate metrics are Hit Ratio (HR) and Normalized Discounted Cumulative Gain (NDCG).

Parameter Settings. In ONFM, the weight of all missing data is set to c_0 uniformly, the batch size is set to 512, the embedding size is set to 64, the learning rate is set to 0.05, and the dropout ratio is set to 0.9. c_0 is set to 0.05 and 0.005 for Frappe and Lisr.fm, respectively.

4.2 Performance Comparison

Table 1 summarize the best performance of these models on Frappe and List.fm, respectively. In order to evaluate on different recommendation lengths, we set the length K = 5, 10, and 20 in our experiments. The experimental results show that our model achieves the best performance on all datasets regarding to both HR and NDCG. ONFM-1 adds the third-order interactions between features based on ENSFM. It is noted that ONFM-1 uses shared embedding. Compared with ENSFM, its performance is better, which indicates the effectiveness of the third-order interactions. On the basis of ONFM-1, ONFM-2 introduces the technique called Order-aware Embedding. The performance is improved, indicating that using order-aware embedding is a better choice. ONFM-3 is the final form of our model, which adds the third-order interactions and the fourth-order interactions meanwhile, and also use Order-aware Embedding. Compared with ENSFM, the performance of ONFM-3 is excellent.

5 Conclusion and Future Work

In this paper, we propose a novel model named Order-Aware Embedding Non-Sampling Factorization Machines. The key design of ONFM is to transform FM model incorporating the high-order interactions into a MF form through mathematical transformation. Then we can get three auxiliary vectors—$\mathbf{p}_u$, $\mathbf{q}_v$ and $\mathbf{h}_{aux}$. $\mathbf{p}_u$ and $\mathbf{q}_v$ are only related to the corresponding user and item. We also use Order-aware Embedding. Finally, through some optimization methods, we apply non-sampling to train ONFM. Extensive experiments on two datasets demonstrate that ONFM obtains effective feature information successfully.

Although the results of ONFM illustrate the importance of the high-order interactions, the way to calculate the high-order interactions is crude. In the future, we will design a more excellent method to calculate the high-order interactions. Moreover, different feature interactions have different influence on the accuracy of the final prediction. So in order to better extract feature information, we are also interested in applying attention mechanism to our model.

References

1. Chen, C., Zhang, M., Ma, W., Liu, Y., Ma, S.: Efficient non-sampling factorization machines for optimal context-aware recommendation. In: WWW (2020)
2. Chen, C., et al.: An efficient adaptive transfer neural network for social-aware recommendation. In: SIGIR (2019)
3. Chen, C., Zhang, M., Zhang, Y., Liu, Y., Ma, S.: Efficient neural matrix factorization without sampling for recommendation. ACM Trans. Inf. Syst. **38**(2), 14:1–14:28 (2020)
4. Guo, H., Tang, R., Ye, Y., Li, Z., He, X.: DeepFM: a factorization-machine based neural network for CTR prediction. In: IJCAI (2017)
5. Guo, W., Tang, R., Guo, H., Han, J., Yang, W., Zhang, Y.: Order-aware embedding neural network for CTR prediction. In: SIGIR (2019)

6. He, X., Chua, T.: Neural factorization machines for sparse predictive analytics. In: SIGIR (2017)
7. He, X., Du, X., Wang, X., Tian, F., Tang, J., Chua, T.: Outer product-based neural collaborative filtering. In: IJCAI (2018)
8. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.: Neural collaborative filtering. In: WWW (2017)
9. Rendle, S.: Factorization machines. In: ICDM (2010)
10. Xin, X., Chen, B., He, X., Wang, D., Ding, Y., Jose, J.: CFM: convolutional factorization machines for context-aware recommendation. In: IJCAI (2019)
11. Xin, X., Yuan, F., He, X., Jose, J.M.: Batch IS NOT heavy: learning word representations from all samples. In: ACL (2018)
12. Yuan, F., et al.: f_{bgd}: learning embeddings from positive unlabeled data with BGD. In: UAI (2018)