MR-FI: Mobile Application Recommendation Based on Feature Importance and Bilinear Feature Interaction

Mi Peng, Buqing Cao(✉), Junjie Chen, Jianxun Liu, and Rong Hu

School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan, China

Abstract. With the rapid growth of mobile applications in major mobile app stores, it is challenging for users to choose their desired mobile applications. Therefore, it is necessary to provide a high-quality mobile application recommendation mechanism to meet the user's expectation. Although the existing methods make significant results on mobile application recommendation, the recommendation accuracy can be further improved. More exactly, they mainly focus on how to better interact between mobile applications' features, but ignore the importance or weight of these features themselves. Based on squeeze-excitation network mechanism and bilinear function with combining inner product and Hadamard product, this paper proposes a mobile application recommendation method based on feature importance and bilinear feature interaction to solve this problem. First of all, it exploits a SENET (Squeeze-Excitation Network) mechanism to dynamically learn the importance of mobile applications' features and uses a bilinear function with combining inner product and Hadamard product to effectively learn these features interactions, respectively. Then, the user preferences for different mobile applications are predicted through infusing cross-combined features into a deep model via integrating the classic deep neural network component with the shallow model. The real dataset of Kaggle is used to evaluate the proposed method and the experimental results show that the method can achieve the best results in most cases in terms of AUC and Logloss. It can effectively improve the recommendation accuracy of mobile applications.

Keywords: Mobile application · Recommendation · Feature importance · Bilinear feature interaction

1 Introduction

With the rapid development of 5G, industrial Internet and mobile network, mobile devices have become a very common and indispensable "intermediary" in people's daily life. According to the 46th statistical report on Internet development in China, up to June 2020, China's users in mobile phone have reached 932 million, in which 99.2% Internet users use mobile phones to access the Internet. With the popularity of smart phones and other mobile devices, the number of mobile applications has shown

H. Gao and X. Wang (Eds.): CollaborateCom 2021, LNICST 406, pp. 213–228, 2021.
https://doi.org/10.1007/978-3-030-92635-9_13

explosive growth [1]. Facing such large number of mobile applications with rich information, it is difficult for users to find suitable mobile applications [2]. Therefore, it is necessary to recommend personalized mobile applications to users.

Recommender system is now playing an important role in helping users out of the mobile service overload predicament and automatically recommending proper mobile applications for users [3]. During the process of mobile applications recommendation, the historical interaction information, such as users' rating or comments, can be exploited to estimate the possibility of selecting different mobile applications for users [4]. Among this, traditional collaborative filtering has been widely adopted [5]. It discovers users' preferences for mobile applications by mining user's historical behavior data, and the mobile applications that users may like are predicted for recommendation. That is to say, collaborative filtering achieves the function, such as "guess what you like", "the people who used the mobile application also like". Due to the problem of sparsity and scalability, the recommendation performance of collaborative filtering can be improved. MF (Matrix Factorization) is an enhanced collaborative filtering with collective wisdom and implicit semantics, which models the interaction between users and mobile applications with inner product [6]. It maps user-mobile application scoring matrix with high-dimension into two user-mobile application matrices with low-dimension to solves the problem of data sparsity. Moreover, some extended models on top of matrix factorization are proposed. For example, NCF (Neural Collaborative Filtering) exploits the nonlinear neural network to replace the interaction function of inner product in matrix factorization [7], CDL (Collaborative Deep Learning) extends the embedding function of matrix factorization by combining the deep representations of rich side (feature) information in mobile applications [8]. Most recently, fine-grained models in mobile application recommendation on the basis of matrix factorization via discriminating low-order features, high-order features and their importance are investigated. For instance, NFM (Neural Factorization Machine) [9] and DeepFM (Deep Factorization Machine) [10] simultaneously consider the low-order and high-order feature interactions, MLR (Multiple Linear Regression) [11] only exploits the high-order feature interactions, and AFM (Attentional Factorization Machine) emphasizes the importance (weight) of feature interactions between users and mobile applications [12].

Although the above models and methods make significant results in recommender system, the recommendation accuracy can be further improved. Specifically, they mainly focus on how to better interact between features, but ignore the importance or weight of features themselves. In fact, users have different preferences for different features in mobile applications. For example, compared with the price and size of mobile applications, most users care more about the rating of mobile applications. In other words, different features have different importance to the task of mobile application recommendation. In view of this, inspired by the work of Huang et al. [13], this paper proposes a mobile application recommendation method based on feature importance and bilinear feature interaction, abbreviated as MR-FI. It exploits a SENET (Squeeze-Excitation Network) mechanism to dynamically learn the importance of features, and uses a bilinear function with combining inner product and Hadamard product to effectively learn the features interactions [14], respectively. In summary, the contributions of this paper are as follows:

- To our best knowledge, this is the first work to introduce SENET mechanism into mobile application recommendation, which is conducive to mine the importance of features in mobile applications.
- We propose a novel mobile application recommendation method based on feature importance and bilinear feature interaction, via employing the SENET to learn the importance of features dynamically and the three types of bilinear interaction layer to learn feature interactions in a fine-grained way.
- Through the real dataset of Kaggle, we verify the effectiveness and accuracy of the SENET mechanism used in mobile application recommendation. The experimental results indicate the performance of MR-FI is superior to that of all the comparison models in most cases in terms of AUC and Logloss.

The remainder of this paper is arranged as follows: the Sect. 2 is the related work of mobile application recommendation; the Sect. 3 is the specific proposed method; the Sect. 4 is the experimental evaluation and analysis; and the last section is the summary of our work and the follow-up research work.

2 Related Work

It is challenging for users to find their own interesting mobile applications quickly from a large number of mobile applications with rich content. Therefore, the relevant scholars study mobile application recommendation [17, 25], in order to improve the efficiency of mobile application searching and the accuracy of recommendation results. The main method in mobile application recommendation is based on content information and network structure.

Content-based mobile application recommendation mainly recommends mobile applications to users through user's text description, tags, user information, etc. [17, 21]. For example, Chen et al. [17] propose a novel third-party library recommendation method by integrating topic modeling and knowledge graph technology, which extracts topics from text application description, and uses knowledge graph to merge structured information of third-party library and application, as well as interactive information of recommended application and library. Cao et al. [18] use app information and user information of multiple platforms to improve the accuracy of mobile application recommendation. This alleviates the problem of data sparsity and cold start to a certain extent. Zhong et al. [19] design a mobile application recommendation method based on hyperlinks induced topic search (hits) algorithm and association rules, which considers the importance of mobile applications in association rules and the reliability of users. Pai et al. [20] calculate the positive and negative scores of semantic tendency in each comment according to the pointwise mutual information, and consider the subjective factors such as public opinion, anonymous opinion and star rating, and the objective factors such as the number of downloads and reputation. Xu et al. [21] present an effective function extraction method. One of the main features of this work is to extract mobile application functions from user reviews for recommendation.

Mobile application recommendation based on network structure exploits the similarity between mobile applications to build a network and recommend related mobile

applications to users [22, 25]. For example, Liang et al. [22] propose to model functional interaction from different views through attention mechanism. The novelty of this method is that it introduces view segmentation for feature interaction and two levels of attention network construction. Xie et al. [23] use user's historical behavior data and application's auxiliary information to design application recommendation to solve the problem of information overload. Cheng et al. [24] utilize a mechanism to model the three important factors that control the application installation of smartphone users. They may classify the application to be installed as one of these factors, and provide application suggestions for users accordingly. Donghwan et al. [25] express the user's usage pattern as a graph. Based on the graph, the user's recommended value for unknown applications is predicted by measuring the similarity.

In recent years, many applications of CTR model based on deep learning to recommendation system are proposed. FNN [15] supported by factor decomposing machine is a forward neural network which uses FM to pre-train embedded layer. However, FNN can only capture higher-order feature interaction. The wide&deep model (WDL) [16] is introduced in google-play for application recommendation. WDL combines training of wide linear model and deep neural network to recommend the system based on the advantages of memory and generalization. However, the input of WDL in wide domain still needs professional feature engineering, which means that cross domain transformation also needs manual design. In order to reduce the manual work in feature engineering, DeepFM [10] replaces the wide part of WDL with FM, and shares feature embedding between FM and deep components. DeepFM is one of the most advanced models in CTR estimation field. NFM [9] combines the second-order linear feature extracted by FM and the higher-order nonlinear feature extracted by neural network, and NFM has more expressive capability than FM. As mentioned in [12], FM can be hindered by its modeling of all feature interactions with the same weight, because not all feature interactions have the same usefulness and predictability. They propose the attention factor decomposing machine (AFM) [12] model, which can learn the importance of each feature interaction automatically from the data through neural attention network, so as to realize the different contribution of feature interaction to prediction. MLR can be regarded as a natural extension of LR. It uses the idea of dividing and governing, and uses piecewise linear model to fit the nonlinear classification surface of high-dimensional space.

3 Proposed Method

The goal of MR-FI is to dynamically learn features and the importance of feature interaction in a more fine-grained way. The structure of the model is shown in Fig. 1. The MR-FI model consists of embedding layer, SENET layer, bilinear-interaction layer, connectivity layer, neural network and prediction layer. Among them, the embedding layer firstly embeds the sparse representation of the original input features into the dense vector. Secondly, the SENET layer transforms the embedding layer into the SENET-like embedded features to improve the distinguishability of the features. Thirdly, the bilinear-interaction layer models the original embedding and the SENET-like embedding by the second-order feature interaction. Fourthly, the connectivity layer

integrates the output connection of the bilinear interactive layer into the dense vector. Finally, the cross-combined features are infused into the neural network, and the prediction score is achieved in the prediction layer.

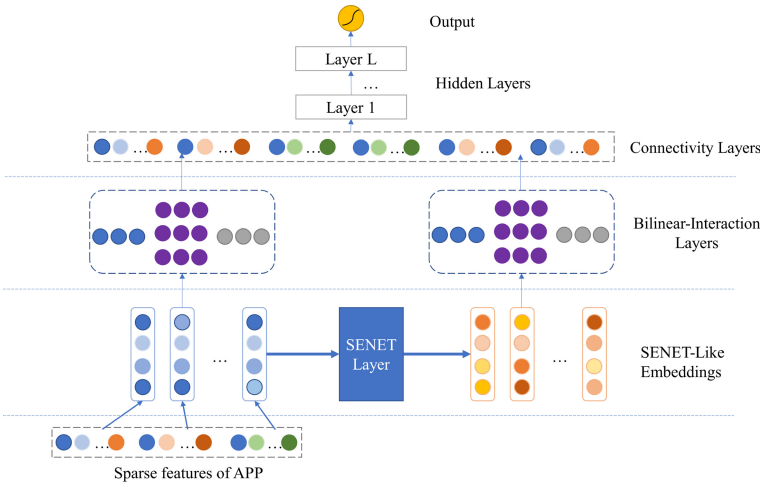

Fig. 1. Structure description of MR-FI model.

3.1 Embedding Layer

The sparse input layer uses sparse representation for the original input features. The embedding layer can embed the sparse features into the low dimensional continuous real valued vector, transform the sparse matrix into the dense matrix by linear transformation, extract the hidden features of the matrix, and improve the generalization ability of the model. The output of the embedded layer is expressed as follows:

$$E = \left[e_1, e_2, .., e_i, \ldots, e_f\right] \tag{1}$$

Where f is the number of domains, $e_i \in \mathbb{R}^k$ is the representation of the i-th domain, and the size of the vector is k. Sparse input layer and embedded layer are widely used in deep learning-based hit rate prediction models, such as NFM and FNN.

3.2 SENET Layer

Different features have different importance to the target task. For example, when predicting a user's preference for a mobile application, the score of the mobile application may be more important than the price of the mobile application. Inspired by the successful application of SENET [14] in the field of computer vision, this paper introduces the SENET mechanism to make the model pay more attention to the importance of features. For a specific mobile application recommendation task, we can

dynamically increase the weight of important features and reduce the weight of features with insufficient information through the SENET mechanism.

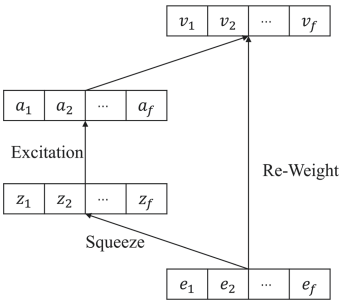

Fig. 2. The SENET layer.

As Fig. 2 illustrated, the SENET consists of three steps: (1) Squeeze, (2) Excitation, (3) Re-Weight. Firstly, the global feature is obtained by the squeeze operation on the embedding feature obtained in the embedding layer, and then the excitation operation is performed on it to learn the relationship between each embedding, and obtain the weight of different domain embedding. Finally, the final embedding result is obtained by multiplying the original embedding. The specific steps are as follows:

Squeeze. The first step is to compress the original embedding E into the statistical vector $Z = [z_1, \ldots, z_i, \ldots, z_f]$ by using the average pooling operation. z_i can be calculated by the following formula:

$$z_i = \frac{1}{k} \sum\nolimits_{t=1}^{k} e_i^{(t)} \tag{2}$$

Where, z_i is the global information about the i-th feature representation, and k is the embedded dimension size.

Excitation. The second step is to learn the embedding weights of each domain based on the statistical vector Z, and use two fully connected (FC) layers to learn the weights. The first fully connected layer is the dimension reduction layer of parameter W_1, which uses the σ_1 as a nonlinear function. The second fully connected layer restores the original dimension by increasing the dimension with parameter W_2. Formally, the weight of domain embedding can be calculated as follows:

$$A = \sigma_2(W_2\sigma_1(W_1 Z)) \tag{3}$$

Where $A \in \mathbb{R}^f$ is the vector, σ_1 and σ_2 is the activation function.

Re-Weight. The last step is to re-weight, that is, each field of the embedding layer is multiplied by the corresponding weight, and the final embedding result is $V = \{v_1, \ldots, v_f\}$. The whole operation can be seen as learning the weight co-efficient

of each domain embedding, which makes the model more discriminative to each domain embedded features. The embedded V of class SENET can be calculated as follows:

$$V = [a_1 \cdot e_1, \ldots, a_f \cdot e_f] = [v_1, \ldots v_f] \tag{4}$$

In short, the SENET uses two fully connected layers to dynamically learn the importance of features. For the recommendation task of mobile applications, it increases the weight of important features and reduces the weight of features with insufficient information.

3.3 Bilinear-Interaction Layer

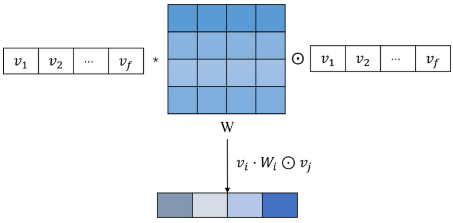

Fig. 3. The method to calculate the feature interactions.

The interaction layer is a layer to calculate the second-order feature interaction. The classical methods of feature interaction are inner product and Hadamard product. The inner product $(\cdot)$ is widely used in shallow models such as FM and FFM, while Hadamard product $(\odot)$ is commonly used in deep models such as NFM. For example, the inner product is $[a_1, a_2, \ldots, a_n] \bullet [b_1, b_2, \ldots, b_n] = \sum_{i=1}^{n} a_i b_i$. The Hadamard product is $[a_1, a_2, \ldots, a_n] \odot [b_1, b_2, \ldots, b_n] = [a_1 b_1, a_2 b_2, \ldots, a_n b_n]$. Because the inner product and Hadamard product of the interaction layer are too simple to model the feature interaction of sparse data sets effectively. Therefore, this paper uses a more fine-grained method, combining inner product and Hadamard product to learn feature interaction with additional parameters. As Fig. 3 illustrated, the interaction vector p_{ij} can be calculated in the following three ways:

Field-All Type

$$p_{ij} = v_i \cdot W \odot v_j \tag{5}$$

Where $W \in \mathbb{R}^{k \times k}$, all vectors (v_i, v_j) share the W. So, it is called as "Field-All". The additional parameter is $k \times k$.

Field-Each Type

$$p_{ij} = v_i \cdot W_i \odot v_j \tag{6}$$

Among them, $W_i \in \mathbb{R}^{k \times k}$ corresponds to the parameter matrix of the i-th field, so this layer needs to maintain $f \times k \times k$ parameters, which is called as "Field-Each".

Field-Interaction Type

$$p_{ij} = v_i \cdot W_{ij} \odot v_j \tag{7}$$

Among them, $W_{ij} \in \mathbb{R}^{k \times k}$ corresponds to the weight matrix between the i-th domain and the j-th domain. In addition, it needs an additional parameter of $\frac{f(f-1)}{2} \times k \times k$, which involves more parameter matrices. Therefore, it is the most time-consuming calculation method among the three methods, which can improve the accuracy of the recommended model to a certain extent.

In this part, the bilinear-interaction layer can output the interaction vector $p = [p_1, \ldots, p_i, \ldots, p_n]$ from the original embedding E and interaction vector $q = [q_1, \ldots, q_i, \ldots, q_n]$ from the class SENET embedding V.

3.4 Connectivity Layer

The connectivity layer connects the interaction vectors p and q, and infuses the connected vectors into the next layer of MR-FI model, a standard neural network layer. The specific process can be expressed as follows:

$$C = [p_1, \ldots, p_n, q_1, \ldots, q_n] = [c_1, \ldots, c_{2n}] \tag{8}$$

If each element of vector C is summed up and a sigmoid function is used to output a predictive value, a shallow prediction model is gotten. In order to further improve the performance of the model, this paper considers the combination of shallow component and deep neural network (DNN).

3.5 Deep Network

In order to further improve the performance, we combine the classic deep neural network (DNN) component with the shallow model to form a deep model. The input of this layer is the output vector C of the connectivity layer, and the deep network is composed of multiple fully connected layers, which can implicitly capture higher-order features. Let $a^{(0)} = [c_1, \ldots, c_{2n}]$ denotes the initial input. Next, $a^{(0)}$ is poured into the deep neural network, and the feedforward process is as follows:

$$a^{(l)} = \sigma\left(W^l a^{(l-1)} + b^l\right) \tag{9}$$

Where $a^{(l)}$ is the output of layer l of the deep network, σ is a function of sigmoid, W^l represents the weight matrix of the model, and b^l represents the offset of the model.

After L layer, a dense real valued eigenvector is generated and input into *sigmoid* function for mobile application prediction.

$$y_d = \sigma\left(W^L a^{(L+1)} + b^{L+1}\right) \tag{10}$$

Where L is the number of layers of the deep model.

3.6 Prediction Layer

Finally, the output expression of the model prediction layer is as follows:

$$\widehat{y} = \sigma\left(w_0 + \sum\nolimits_{i=0}^{m} w_i x_i + y_d\right) \tag{11}$$

Where $\widehat{y} \in (0, 1)$ is the predictive value of the model, σ is the *sigmoid* function, m is the feature size, and w_i is the i-th weight of the linear part.

The whole training process aims to minimize the following objective function:

$$Logloss = -\frac{1}{N}\sum\nolimits_{i=1}^{N}(y_i log(\widehat{y}_i)) + (1 - y_i) \times log(1 - \widehat{y}_i)) \tag{12}$$

Where, y_i is the real label of the i-th mobile application, $\widehat{y}_i$ corresponds to the prediction tag of the i-th mobile application, and N is the total number of mobile applications.

4 Experimental Result and Analysis

4.1 Data Set and Experiment Setup

This experiment uses the open data set "App Store" of kaggle as the experimental data set. In this data set, a total of 11 features are selected, including 2 sparsity features and 9 continuity features, that are prime_ genre, cont_ rating, price, rating_ count_ tot, rating_ count_ ver, user_ rating, user_ rating_ ver, sup_ devices.num, ipadSc_ urls. num, lang.num and size_ MB. Since there is no label in the original data set, we manually set the label value of mobile applications with a score of more than 3 and a score of more than 800 as 1, otherwise are 0. The proportion of positive samples is about 0.424, so that the experimental results will not be affected too much due to the uneven distribution of samples. In the experiment, the optimizer is Adam and the loss function is binary cross-entropy.

During model training, the batch size is set to 64, and the proportion of verification set is 0.2. In order to better illustrate the experimental results, different proportions of

Table 1. Category number statistics of top20 mobile applications.

Category name	Number	Category name	Number
Games	3381	Lifestyle	98
Entertainment	456	Shopping	82
Education	408	Weather	69
Photo & video	339	Book	59
Utilities	202	Travel	58
Health & fitness	166	News	55
Productivity	164	Business	54
Music	136	Reference	53
Social networking	113	Finance	47
Sports	103	Food & drink	45

training sets are selected to test the experimental results. The distribution details of the top 20 categories with the largest number are shown in Table 1.

4.2 Evaluation Metrics

AUC and Logloss, which are widely used in click through prediction, are selected as evaluation indexes [10].

AUC is the area under the *ROC* curve. When $0.5 < AUC < 1$, the model is better than the random classifier. In particular, the closer the *AUC* is to 1.0, the higher the authenticity is; When it is equal to 0.5, the authenticity is the lowest. The calculation formula is as follows:

$$\mathrm{AUC}_i = \int_0^1 \mathrm{ROC}_i(fpr)d(fpr) \tag{13}$$

Where *fpr* stands for false positive rate and *fpr* stands for true positive rate. In *ROC* space, the coordinate point (fpr, fpr) describes the trade-off between *FP (false positive case)* and *TP (true positive case)*.

Logloss measures the accuracy of a classifier by punishing the wrong classification. Minimizing the logarithmic loss is basically equivalent to maximizing the accuracy of the classifier. Logloss reflects the average deviation of samples, and is often used as the loss function of the model to optimize. The calculation formula is as follows:

$$Logloss = -\frac{1}{N}\sum_{i=1}^{N}(y_i log(\widehat{y}_i)) + (1 - y_i) \times log(1 - \widehat{y}_i)) \tag{14}$$

Where, y_i is the real label of the i-th sample, $\widehat{y}_i$ corresponds to the prediction tag of the i-th sample, and N is the total number of mobile application samples.

4.3 Baseline Methods

This paper compares the following methods to better illustrate the experimental results.

MLR [11]: MLR model is a generalization of linear LR model, which uses piecewise linear method to fit data, and can learn higher-order feature combination. The basic strategy is to divide and conquer.

AFM [12]: improves the FM model and introduces the attention mechanism into the feature crossover module. An attention network is used to learn the importance of different combination features (second-order crossover).

FNN [15]: FNN (neural network based on factor decomposition machine) uses FM for supervised learning to obtain the embedding layer, which can effectively reduce the dimension of sparse features and obtain continuous and dense features.

NFM [9]: by combining FM's linear crossover of second-order features with neural network's nonlinear crossover of higher-order features, NFM learns feature combinations that never appear in the training set.

DeepFM [10]: combines the first-order and second-order features of FM and the interaction of higher-order features, and reduces the amount of parameters and shares information by sharing the embedding of FM and DNN.

4.4 Experimental Performance

In this section, we select the training set ratio from 0.2 to 0.9 to compare the model performance. The experimental results are shown in Table 2, Figs. 4 and 5, respectively. In general, MR-FI has the best performance. Especially when the train_size is 0.8, the AUC of MR-FI is 19.2%, 20.99%, 1.22%, 0.27% and 1.08% higher than that of MLR, AFM, FNN, NFM and DeepFM, respectively. Compared with the above methods, the logloss of MR-FI decreased by 32.73%, 34.72%, 2.22%, 3.37% and 3.28%. More specifically, we have the following findings:

MR-FI has the best performance in all models, which shows that considering the weight of original features and using Bilinear interactive learning feature interaction can effectively improve the accuracy of the recommended model.

The performance of depth models such as FNN and DeepFM is better than that of MLR. When the training size is 0.8, the performance of FNN and DeepFM is improved by 17.98% and 18.21%, respectively. It shows that the depth model can better model when features are sparse.

In the shallow model, MLR is always better than AFM. When the training size is 0.2, the accuracy of MLR model is 4.73% higher than that of AFM model, which indicates that learning the weight of high-order feature combination is more important than that of low-order feature combination.

Table 2. Performance comparison of different proportion training sets.

Models	Train_size = 0.2		Train_size = 0.5		Train_size = 0.8		Train_size = 0.9	
	AUC	Logloss	AUC	Logloss	AUC	Logloss	AUC	Logloss
MLR	0.7343	0.5933	0.7708	0.5604	0.7821	0.5430	0.7959	0.5409
AFM	0.6870	0.6288	0.7529	0.5853	0.7642	0.5629	0.7773	0.5607
FNN	0.8684	0.4811	0.9265	0.4079	0.9619	0.2379	0.9732	0.2538
NFM	0.8736	0.4516	0.9348	0.3384	0.9714	0.2494	0.9739	0.2382
DeepFM	0.8703	0.4584	0.9267	0.3723	0.9633	0.2485	0.9761	0.2068
MR-FI	0.8749	0.4491	0.9328	0.3959	0.9741	0.2157	0.9807	0.1825

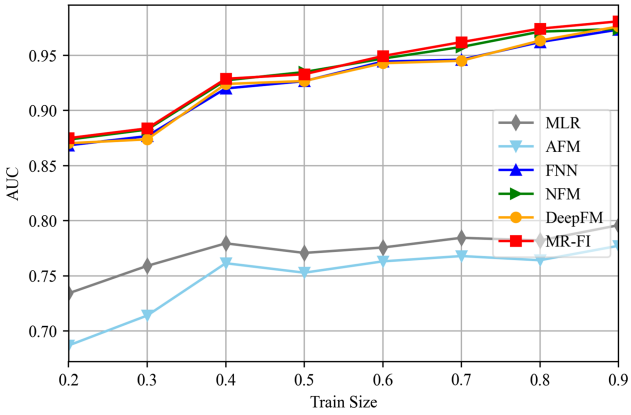

Fig. 4. AUC comparison of different models.

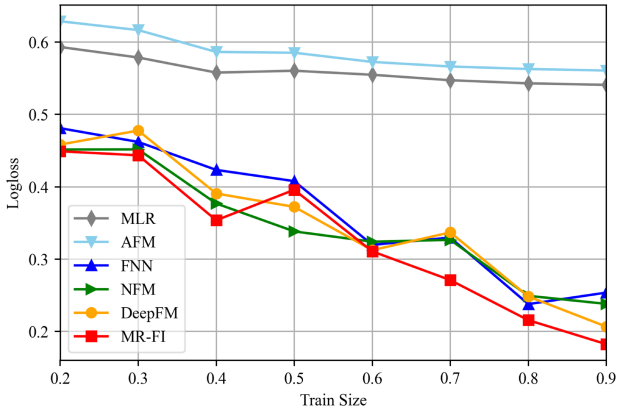

Fig. 5. Logloss comparison of different models.

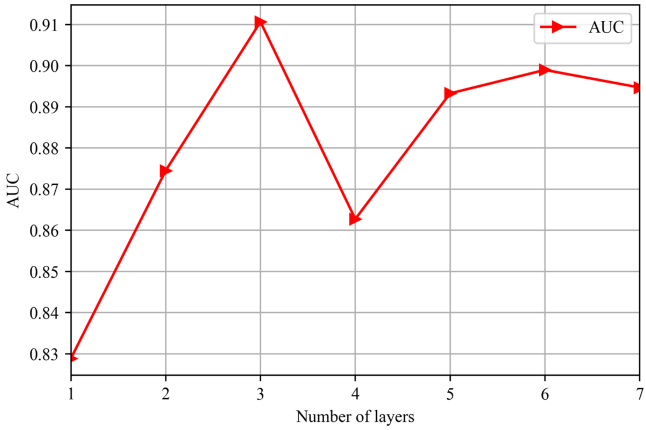

Fig. 6. Influence of neural networks with different layers on AUC.

Starting from FNN, the accuracy of this kind of model is significantly improved by adding neural network. The results show that the neural network can learn more effectively from FM representation, and can get better results than shallow model.

The overall performance of NFM model and DeepFM model is also good, which indicates that the interaction of low-order features and high-order features can improve the accuracy of the model to a certain extent.

4.5 Hyperparameters Analysis

In this part, the hyperparameters analysis mainly includes the size of embedding and the number of layers of neural network. However, in the feature modeling of mobile application recommendation, there are more continuity features and less sparsity features. Therefore, the embedding size has little effect on the accuracy of the model, which is not discussed here. This section mainly analyzes the influence of neural network layers on the model. According to the experimental results of Figs. 6 and 7, it can be observed that: (1) when the number of neural network layers increases from 1 to 3, AUC has been improved to a certain extent, from 0.83 to 0.91; (2) when the number of layers increases to 4, AUC decreases sharply; (3) when the number of neural network layers increases from 4 to 7, AUC has been improved to a certain extent, but it has not reached the highest point of layer 3, and has a downward trend. From the influence of different neural network layers on Logloss, we can find the same trend, that is, when the layer number is 3, Logloss reaches the lowest point. Therefore, we can

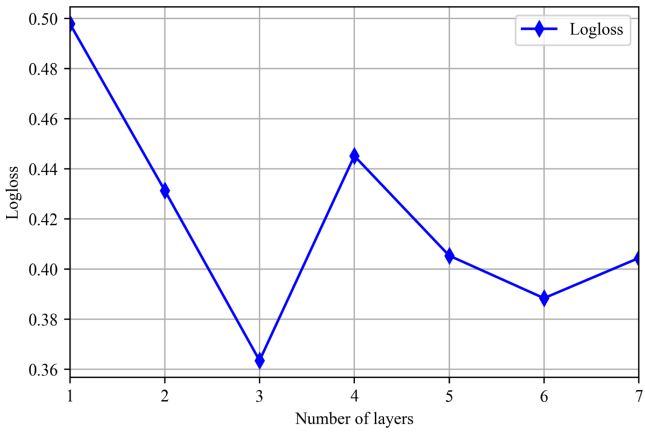

Fig. 7. Influence of neural networks with different layers on logloss.

infer that increasing the number of DNN layers will raise the complexity of the model. Specifically, in a certain number of layers, increasing the number of layers will improve the performance of the model; but if the number of layers increases, subsequently the performance will decline. This is because too complex model can easily lead to overfitting. In particular, for mobile application recommendation, a good choice is to set the layer size as 3.

5 Conclusion and Future Work

In this paper, a mobile application recommendation method based on feature importance and bilinear feature interaction MR-FI is proposed to dynamically learn the feature importance and fine-grained feature interaction of mobile applications. In this method, on the one hand, a more complex bilinear interaction layer is introduced to learn feature interaction, rather than simply using inner product or Hadamard product to calculate feature interaction. On the other hand, for mobile application recommendation tasks, it uses the SENET module to dynamically learn the importance of features, which will enhance the weight of important features and reduce the weight of unimportant features. By combining SENET mechanism with bilinear feature interaction, the MR-FI model can further improve the accuracy of mobile application recommendation. Through the verification on the open data set of Kaggle, the MR-FI model performs best in most cases. In the future work, we will explore more fine-grained feature interaction and discriminate their importance with different dimensions for further improving the accuracy of mobile application recommendation.

Acknowledgement. Our work is supported by the National Key R&D Program of China (2018YFB1402800), the National Natural Science Foundation of China (No. 61873316, 61872139, 61832014 and 61702181), and the Natural Science Foundation of Hunan Province (No. 2021JJ30274).

References

1. Gao, S., Zang, Z., Gopalakrishnan, S.: A study on distribution methods of mobile applications in China. In: Fong, S., Pichappan, P., Mohammed, S., Hung, P., Asghar, S. (eds.) Seventh International Conference on Digital Information Management, pp. 375–380 (2012)
2. Deng, S., et al.: Toward mobile service computing: opportunities and challenges. IEEE Cloud Comput. **3**, 32–41 (2016)
3. Zhang, D., Lee, W.S.: Question classification using support vector machines. In: Clarke, C. L.A., Cormack, G.V., Callan, J., Hawking, D., Smeaton, A.F. (eds.) SIGIR 2003: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, New York, pp. 26–32 (2003)
4. Deerwester, S.C., Dumais, S.T., Landauer, T.K., Furnas, G.W., Harshman, R.A.: Indexing by latent semantic analysis. J. Am. Soc. Inf. Sci **41**(6), 391–407 (1990)
5. Yang, Z., Wu, B., Zheng, K., Wang, X., Lei, L.: A survey of collaborative filtering-based recommender systems for mobile internet applications. IEEE Access. **4**, 3273–3287 (2016)
6. Koren, Y., Bell, R.M., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer **42**, 30–37 (2009)
7. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.S.: Neural collaborative filtering. In: Proceedings of the 26th International Conference on World Wide Web, pp. 173–182 (2017)
8. Wang, H., Wang, N., Yeung, D.Y.: Collaborative deep learning for recommender systems. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1235–1244 (2015)
9. He, X., Chua, T.S.: Neural factorization machines for sparse predictive analytics. In: Kando, N., Sakai, T., Joho, H., Li, H., Vries, A.P. de, and White, R.W. (eds.) Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, pp. 355–364 (2017)
10. Guo, H., Tang, R., Ye, Y., Li, Z., He, X.: DeepFM: A Factorization-machine based neural network for CTR prediction. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence, pp. 1725–1731 (2017)
11. Gai, K., Zhu, X., Li, H., Liu, K., Wang, Z.: Learning piece-wise linear models from large scale data for Ad click prediction. CoRR. abs/1704.05194 (2017)
12. Xiao, J., Ye, H., He, X., Zhang, H., Wu, F.: Attentional factorization machines: learning the weight of feature interactions via attention networks. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence, pp. 3119–3125 (2017)
13. Huang, T., Zhang, Z., Zhang, J.: FiBiNET: combining feature importance and bilinear feature interaction for click-through rate prediction. In: Bogers, T., Said, A., Brusilovsky, P., and Tikk, D. (eds.) Proceedings of the 13th ACM Conference on Recommender Systems, pp. 169–177 (2019)
14. Hu, J., Shen, L., Albanie, S., Sun, G., Wu, E.: Squeeze-and-excitation networks. IEEE Trans. Pattern Anal. Mach. Intell. **42**(8), 2011–2023 (2020)
15. Zhang, W., Du, T., Wang, J.: Deep learning over multi-field categorical data. In: Ferro, N., et al. (eds.) ECIR 2016. LNCS, vol. 9626, pp. 45–57. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-30671-1_4
16. Cheng, H.-T., et al.: Wide and deep learning for recommender systems. In: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, pp. 7–10 (2016)
17. Chen, J., Li, B., Wang, J., Zhao, Y., Yao, L., Xiong, Y.: Knowledge graph enhanced third-party library recommendation for mobile application development. IEEE Access **8**, 42436–42446 (2020)

18. Cao, B., Chen, J., Liu, J., Wen, Y.: A topic attention mechanism and factorization machines based mobile application recommendation method. Mobile Netw. Appl. **25**(4), 1208–1219 (2020). https://doi.org/10.1007/s11036-020-01537-z
19. Pai, H.-T., Lai, H.W., Wang, S., Wu, M.F., Chuang, Y.T.: Recommendations for mobile applications: facilitating commerce in google play. In: Hamdan, H., Boubiche, D.E., and Klett, F. (eds.) Proceedings of the 1st International Conference on Internet of Things and Machine Learning, pp. 10:1–10:6. ACM (2017)
20. Pai, H.T., Lai, H.W., Wang, S., Wu, M.F., Chuang, Y.T.: Recommendations for mobile applications: facilitating commerce in google play. In: Hamdan, H., Boubiche, D.E., and Klett, F. (eds.) Proceedings of the 1st International Conference on Internet of Things and Machine Learning, pp. 10:1–10:6 (2017)
21. Xu, X., Dutta, K., Datta, A.: Functionality-based mobile app recommendation by identifying aspects from user reviews. In: Myers, M.D., Straub, D.W. (eds.) Proceedings of the International Conference on Information Systems - Building a Better World through Information Systems, ICIS, Auckland, New Zealand (2014)
22. Liang, T., Zheng, L., Chen, L., Wan, Y., Yu, P.S., Wu, J.: Multi-view factorization machines for mobile app recommendation based on hierarchical attention. Knowl. Based Syst. **187**, 104821 (2020)
23. Xie, F., Cao, Z., Xu, Y., Chen, L., Zheng, Z.: Graph neural network and multi-view learning based mobile application recommendation in heterogeneous graphs. In: 2020 IEEE International Conference on Services Computing, Beijing, China, pp. 100–107 (2020)
24. Cheng, V.C., Chen, L., Cheung, W.K., Fok, C.-K.: A heterogeneous hidden Markov model for mobile app recommendation. Knowl. Inf. Syst. **57**(1), 207–228 (2017). https://doi.org/10.1007/s10115-017-1124-3
25. Bae, D., Han, K., Park, J., Yi, M.Y.: AppTrends: a graph-based mobile app recommendation system using usage history. In: 2015 International Conference on Big Data and Smart Computing, pp. 210–216. IEEE Computer Society, Jeju, Korea (2015)

The Missing POI Completion Based on Bidirectional Masked Trajectory Model

Jun Zeng(✉), Yizhu Zhao, Yang Yu, Min Gao, and Wei Zhou

School of Big Data and Software Engineering, Chongqing University, Chongqing, China
{zengjun, zhaoyizhu, yuyang96, gaomin, zhouwei}@cqu.edu.cn

Abstract. With the development of location-based social networks (LBSNs), users can check in Point-of-Interest (POIs) at any time. However, users do not check in all places they have visited, so the POI trajectory sequence generated through LBSNs is incomplete. An incomplete POI trajectory will have a negative impact on subsequent tasks such as POI recommendation and next POI prediction. Therefore, we complete the missing POI in the user trajectory sequence. Since the POI trajectory sequence is incomplete, it is a challenge to use the pre-order and post-order trajectory sequences with missing POIs. Therefore, we propose a masked POI trajectory model (MPTM) that uses the bidirectionality of BERT to complete the missing POIs in user's behavior sequence. By masking the missing POIs, MPTM fully explores the relationship between the missing POIs and the known POIs to predict the missing POIs. In order to strengthen the relationship between POIs in the user trajectory sequence, we build a graph for each user's incomplete POIs sequence to explore the user's hidden behavior habits. Besides, we design experiments to explore the relationship between the continuity of the number of missing POIs and the predictive ability of the model. The experimental results demonstrate that our MPTM outperforms the state-of-the-art models for completion on missing POIs of user's behavior sequence.

Keywords: The missing POIs · Missing POIs completion · Bidirectional masked trajectory model

1 Introduction

With the rapid development of information technology, human mobility behavior is more easily digitized and shared with friends [1]. Especially with the rapid growth of Location-Based Social Networks (LBSNs), such as Yelp, Gowalla and Foursquare. Point-of-Interest (POI) research has attracted wide attention from both academia and industry [2, 3]. However, users do not check in all places they have visited, there are some missing POIs in the user trajectory data collected by LBSN. In reality, the check-in POIs delivered by users is typically incomplete [4]. An incomplete POI trajectory will have a negative impact on subsequent tasks such as POI recommendation and next POI prediction. Therefore, we complete the missing POI in the user trajectory sequence. Existing studies are mainly focused on next location prediction or POI

H. Gao and X. Wang (Eds.): CollaborateCom 2021, LNICST 406, pp. 229–243, 2021.
https://doi.org/10.1007/978-3-030-92635-9_14

recommendation [5, 6]. POI recommendation is to analyze all the historical check-in data of the user and dig out its internal connections, and predict the user's next check-in location to complete the recommendation. While the missing POI completion is to learn the user's historical check-in data and complete the missing POI. This requires bidirectional learning of the user's POI trajectory sequence. However, discovering and integrating the user's behavior sequence relationship for the completion of the missing POIs in sequence is challenging. The reason is that it is difficult to learn the context of missing positions in the user sequence and establish the connection between POIs due to their incompletion.

The current research is mainly for the completion of GPS trajectory [7, 8], but there is few research on POI trajectory completion. The current POI research focuses on POI recommendation, next POI prediction, etc. Due to the remarkable achievements of deep learning in the field of POI research, deep learning technology such as RNN [9] has gradually replaced simple forms of Collaborative Filtering (CF) [10]. In the missing POI completion problem, we need to learn the pre-order and post-order trajectory sequences with missing POIs. The above methods cannot solve the problem.

To address the limitations mentioned above, we propose a bidirectional model to learn the representations for users' behavior sequences and complete user's missing POIs in sequence. The bidirectional model is more suitable than unidirectional models in modeling user behavior sequences since all POIs in the bidirectional model can leverage the contexts from both left and right side [11]. We use the bidirectionality of BERT [12] to fully mine the before and after information of the missing POI. The pre-training task masked language model (Mask LM) of BERT combines sentence context information through self-attention to predict masked words. Therefore, we apply this idea to the problem of missing POIs completion in trajectory sequence. We regard the user's POI trajectory sequence as a paragraph, and few words or a sentence are predicted through the Masked LM task. Specifically, inspired by the success of BERT4Rec [11] in users' behavior sequences, we propose applying the Masked LM in the problem of missing POIs completion in users' behavior sequences. Transformer [13] encoder is used to learn the sequence relationship and long-distance information dependence around the masked POI.

Besides, in order to strengthen the connection between POIs, we fully explore the implicit features of POIs checked in by user. A graph is created for each user of checked in POIs and used DeepWalk [14] for vector representation learning. The user's POIs network graph contains the precedence and potential contacts of checked in POIs by users. This can be seen as a hidden habit of the user check-ins.

The contributions of our paper are as follows:

- We propose and solve the problem of the missing POIs in the user trajectory sequence and complete them. The incomplete POI trajectory sequence has negative impact on subsequent tasks such as POI recommendation, location prediction, and human mobility. And completing the missing POIs is very important in epidemic prevention and control.
- In order to solve the association between missing POI and the information before and after the missing position in the sequence, we propose a bidirectional mask POI trajectory model (MPTM). It is combined with graph features to mine the

relationship between the missing POI and the known POI in the trajectory sequence and the characteristics of the user's behavior sequence.
- In order to strengthen the relationship between POIs, we build a graph structure for each user to learn the graph features of POI, and dig out the user's hidden behavior habits.

The rest of the paper is organized as follows. Section 2 summarizes the related work, which is highly relevant to our research. Section 3 is the problem statement of POI trajectory sequence completion. Section 4 provides detailed methodology of our proposed model. Section 5 presents experiments and the results, and Sect. 6 concludes this paper and outlines prospects for future study.

2 Related Work

The current research mainly focuses on POI recommendation, next location prediction, etc. In the POI recommendation problem, Han et al. [15] divided the context information of POI into two groups, namely global and local contexts, and developed different regularization terms to merge them for recommendation. In the next location prediction problem, Zhao et al. [16] propose a new spatio-temporal gated network by enhancing long-short term memory network, where spatio-temporal gates are introduced to capture the spatio-temporal relationships between successive check-ins.

These methods have achieved good results on the corresponding problems in POI research. However, there is currently no way to solve the problem of missing POIs in the user's trajectory sequence and to complete the missing POIs. Zhang et al. [4] elaborate that the check-in information delivered by users is typically incomplete in reality. But their work only alleviates the impact of incomplete POI trajectories on recommendation, and do not solve the problem of missing POIs. Incomplete POI trajectory information has negative impact on subsequent tasks such as POI recommendation, location prediction, and human mobility. Therefore, it is necessary to use mature methods to predict the missing POIs in the POI trajectory sequence and complete it.

Liu et al. [17] are the first to learn from natural language processing, treating each POI as a word, and each user's check-in record as a sentence. Then, train the implicit representation vector of each POI and explore the influence of the temporal implicit representation vector on it. With the successful application of RNN in sequential data modeling, RNN can be used to model the user's access sequence in continuous POI recommendations [16]. And subsequent POI research generally use RNN and its variants [18, 19]. In the missing POI completion problem, bidirectional model is required to learn optimal representations for user POI trajectory sequences. Recently, an attention-based sequence-to-sequence method, Transformer [13], achieved state-of-the-art performance and efficiency on machine translation tasks which had previously been dominated by RNN-based approaches [20, 21]. Specifically, due to the success of BERT [12] Mask LM in text understanding, we consider applying the deep bidirectional multi-headed attention to solve the problem of missing POIs completion.

3 Problem Statement

3.1 Scene Description

When an epidemic breaks out, prevention and control are very important. Information technology can help the country control the epidemic. But when the information is incomplete, it will increase the risk of the epidemic in some areas. As shown in Fig. 1, when a confirmed patient is found, it is necessary to track and investigate the patient's historical trajectory within a week to reduce the risk of the epidemic. The patient has been to the supermarket two days before the diagnosis, but has not checked in location information on the LBSNs. This supermarket is frequently visited by the user and has check-in records in LBSNs within one month. Therefore, there is no supermarket in the patient's historical trajectory in the last week on the LBSNs. When tracking and investigating the patient's historical trajectory in the last seven days through LBSNs, the supermarket will be ignored. People who come into contact with the patient in the supermarket are potentially at risk. The risk of the epidemic in the area where the supermarket is located will increase. Therefore, it is very important to predict and complete the user's missing location information in the prevention and control of the epidemic.

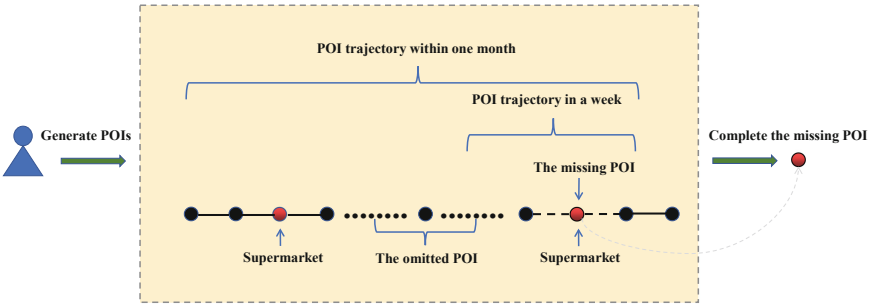

Fig. 1. Scene description

3.2 Problem Definition

In the missing POIs completion problem, the client that uses the LBSNs is called the user, and the generated check-in locations is called the POI. let U and L represent the user and location set respectively. In the POI trajectory sequence generated by the user, the user set is $U = \{u_1, u_2, \ldots, u_{|u|}\}$, the location set is $L = \{l_1, l_2, \ldots, l_{|l|}\}$, and the user POI trajectory sequence $S = \left(l_1^u, l_2^u, \ldots, l_{k-1}^u, l_k^u, l_{k+1}^u \ldots, l_n^u\right)$. User has some missing POIs $l_{missing}^u$. For example, we use $l_{missing}^u$ instead of the missing POI l_k^u. The user's incomplete POI trajectory sequence is $S_m = \left(l_1^u, l_2^u, \ldots, l_{k-1}^u, l_{missing}^u, l_{k+1}^u \ldots, l_n^u\right)$. The model completes the missing supermarkets $l_{missing}^u$ in the last week by learning the user's habit of check-in POI within a month. If the model predicts that the value of $l_{missing}^u$ is the same as the value of l_k^u, the purpose of missing POI completion is achieved.

4 Methodology

4.1 Model Architecture

Different from other POI research problems, the missing POI completion needs to jointly adjust the context in all layers to train deep bidirectional representation. This is because in the problem of missing POI completion, predicting the missing POI needs to combine its surrounding information. Inspired by [12], we use bidirectional masked POI trajectory model (MPTM) to solve the problem of combining the pre-order and post-order trajectory sequences with missing POIs. The Transformer [13] encoder is used to construct a bidirectional MPTM to consider the contextual information around the missing POI. The model structure graph is shown in the Fig. 2.

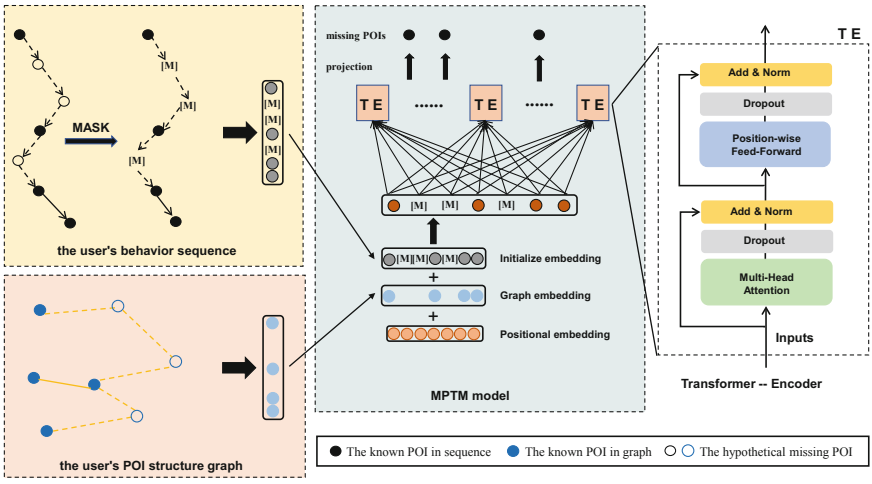

Fig. 2. Model architectures

4.2 Data Processing

For the POI trajectories generated by each user, the POI trajectories are sorted in time to form a sequence. We transform the POI trajectory sequence into a fixed-length n. This fixed-length is determined by the density of the length of the POI checked in by the users. The user's incomplete POI trajectory sequence is $S_m = \left(l_1^u, l_2^u, \ldots, l_k^u, l_{missing}^u, l_{k+1}^u \ldots, l_n^u\right)$, These missing POIs $l_{missing}^u$ are replaced with [M], which are the POI to be predicted by the model. [M] is masked, and refers to mask the missing POIs in the POI trajectory sequence. Therefore, the processed POI trajectory sequence is $S_M = \left(l_1^u, l_2^u, \ldots, l_k^u, [M], l_{k+1}^u \ldots, l_n^u\right)$.

Different from left-to-right language model training, our goal is to let the representation merge the left and right sides of the context to train a deep bidirectional model. Therefore, as same with BERT [12], three methods are used for masking. The

three methods are to replace the original POI with [M], original POI and random POI. First, randomly select 15% of the POI in the POI trajectory sequence. Among these selected POIs, 80% is represented by [M], 10% is represented by original POI, and 10% is represented by random POI. This ensures that the POI trajectory sequence of each input is distributional contextual representation. It can not only let the model know which POIs should be predicted, but also weaken the impact of information leakage of the POI replaced by [M].

4.3 POI Graph of Users

To help the model learn more potential relationships between known POIs and missing POIs, we create a POI trajectory structure graph for each user. The graph contains the precedence and potential contacts of the location checked in by users. This can be seen as a hidden habit of the user check-ins. As shown in Fig. 3, there are nodes $V_1 \rightarrow V_2 \rightarrow V_3 \rightarrow V_2 \rightarrow V_4$ in the user POIs graph. V_1 is home, V_2 is the cafe, V_3 is company, and V_4 is garden. It can be seen that the user is used to going to the cafe when going to the company and leaving the company. This is a hidden habit of the user.

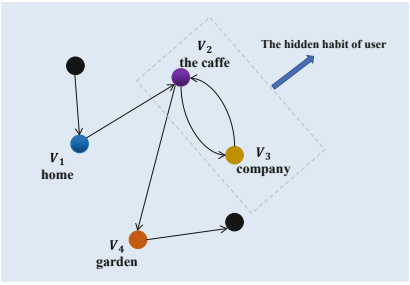

Fig. 3. POI graph of users

We create POI structure graph $G_u = \left(V_{l_1}^u, V_{l_2}^u, \ldots, V_{[M]}, \ldots, V_{l_n}^u\right)$ which are directed edge and weightless graphs for each user. DeepWalk [14] is used to learn the representation of POI in the network, including the topological relationship between POI nodes. DeepWalk can learn a network social representation by truncated random walk. Therefore, the relationship between POIs and the feature vector representation of each POI are obtained. DeepWalk generates a sequence of nodes $S_g = \left(\tilde{l}_1, \tilde{l}_2, \ldots, \tilde{l}_{[M]}, \ldots, \tilde{l}_n\right)$ in the graph through random walk. Where the $\tilde{l}_{[M]}$ is the masked POI. Then learn the latent representation $\tilde{l}_g^u \in R^d$ of the node through local information, where $\tilde{l}_g^u \in L_g^u$. $L_g^u \in R^{n \times d}$ contains the hidden relationship between the POIs that the user check-ins, which can be regarded as the user's hidden habit feature. These features can connect missing POIs with known POIs.

4.4 Embedding Layer

In order to learn the relationship between the known POI and the missing POI, it is necessary to initialize the embedding of the POI to establish the feature dimension. We create a location embedding matrix $L^u \in R^{n \times d}$ based on $S_M = \left(l_1^u, l_2^u, \ldots, l_k^u, [M], l_{k+1}^u \ldots, l_n^u\right)$, where n is the sequence length of user and d is the latent dimensionality. In transformer encoder, there is no iterative operation of Recurrent Neural Networks. All POIs of the sequence into the model for parallel processing at the same time. Therefore, it is necessary to provide position information for each POI in order to infer the missing POI through the relationship between other POIs. Like BERT4rec [11], we also use learnable positional embedding $P \in R^{n \times d}$ for position coding. Besides, we introduce the aforementioned graph feature vector L_g^u, which is combined with the P and L_g^u, and sent to the model together. The added vector contains more features, which can help the model to infer the situational information of missing POI. The combined vector is defined as:

$$In = L^u + P + L_g^u \tag{1}$$

4.5 Transformer-Encoder

Multi-Head Attention

In order to learn the expression of multiple meanings in POI trajectory sequence, it is necessary to perform a linear mapping on the input. The multi-headed attention mechanism can be used to extract the meaning of multiple semantics. The attention function is assigned to each POI weight through the three matrices of $Q(Query), K(key), V(value)$, and the correlation between the known POI and the missing POI is calculated according to the weight. The formulas for multi-head attention is as follows:

$$H \quad (In) = Concat(head_1, head_2, \ldots head_i)W \tag{2}$$

$$head_i = Atttention\left(InW_Q^i, In_iW_K^i, In_iW_V^i\right) \tag{3}$$

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d/h}}\right)V \tag{4}$$

Where weight matrices W_Q, W_K, W_V are generated based on the input, W is a learnable parameter, *Concat* is fully connected, and $head_i$ is the $i-th$ of self-attention. W_Q^i, W_K^i, W_V^i is the feature dimension is divided into h.

Position-Wise Feed-Forward Network

Although multi-head attention can use adaptive weights to aggregate the embedding of known POIs and missing POIs, it is still a linear model. In order to improve the

nonlinearity of the model, and consider the interaction between the missing POI and the known POI in the dimensions. Like Transformer-encoder [13], we also use Position-wise Feed-Forward Network to improve the nonlinearity of the model. The formula is as follows:

$$FFN(H) = GELU(HW_1 + b_1)W_2 + b_2 \tag{5}$$

Where W_1, W_2, b_1, b_2 are all learnable parameters, and $GELU$ is Gaussian Error Linerar Unit.

Stacking Transformer-Encoder Layer
After passing through the multi-head attention module, a network connection is required to learn the potential relationship between the missing POI and other POIs in the input trajectory. We use residual connections to prevent neural network degradation during network training. After each operation of multi-head attention module, the values before and after the operation must be added to obtain the residual connection.

$$Block(H) = LN(A + Drop(FFN(A))) \tag{6}$$

$$A = LN(H + Drop(H)) \tag{7}$$

Where LN is the Layer Normalization and $Drop$ is the Dropout. Layer Normalization can help stabilize the neural network and speed up its training. Dropout can reduce the complex co-adaptation relationship between neurons, and can avoid the phenomenon of over-fitting.

4.6 The Output Layer

After some Transformer-Encoder blocks that adaptively and hierarchically extract information of previous POIs, we get the final output B for all POIs of the input sequence. We need predict the missing POIs base on B. As same as the BERT4Rec, we apply a two-layer feed-forward network with GELU activation to produce an output distribution over target items:

$$O(l) = softmax(GELU(BW_o + b_o)In^T + b) \tag{8}$$

Where W_o is the learnable projection matrix, B is the output after b transformer blocks, b_o and b are bias terms, In is the embedding matrix for the POI set. And the embedding is a matrix shared between the input and output of the model.

4.7 Network Training

Recall that we convert each user POI trajectory sequence to a fixed length sequence $S_m = \left(l_1^u, l_2^u, \ldots, l_k^u, l_{missing}^u, l_{k+1}^u \ldots, l_n^u\right)$ via truncation or padding locations. And the processed user POI trajectory sequence $S_M = \left(l_1^u, l_2^u, \ldots, l_k^u, [M], l_{k+1}^u \ldots, l_n^u\right)$ is generated as matrix vector L^u, which is used as model input with position embedding P and

graph embedding L_g^u. The model outputs the missing POIs representation at the corresponding position, and calculates the loss of the model through the Cross Entropy Loss function. The Cross Entropy Loss is defined as:

$$Loss = -\sum_l (p(l) log q(l)) \tag{9}$$

5 Experiments

5.1 Datasets

We evaluate our proposed method on two real-world LBSN datasets from Foursquare [22], namely NYC and TKY, which have been widely used by previous studies on POI research. This dataset contains check-ins in NYC and Tokyo collected for about 10 months (from 12 April 2012 to 16 February 2013). We remove users with fewer than 10 check-ins and locations which have been visited fewer than 5 times. Table 1 summarizes the statistics of the two datasets.

Table 1. Statistics of datasets.

Datasets	Users	Locations	Check-ins
NYC	1083	38333	227,428
TKY	2293	61858	573,703

Besides, we analyze the distribution and density of user sequence length, as shown in the Figs. 4 and 5. The scatter chart shows that the length of the user's behavior sequence is concentrated within 500 in Fig. 4. And the density map shows the length of user's behavior sequence is concentrated around 140 in Fig. 5. By analyzing these two graphs, we can determine the trend of the centralized distribution of the number of users who checked in the POI. Therefore, the maximum sequence length is set to140. Inspired by SASRec [20], if the sequence length is greater than 140, we consider the most recent 140 actions. If the sequence length is less than 140, we repeatedly add a constant zero vector to the left until the length is 140. We treat the first 50% sequences of each user as training set and validation set, the last 50% as test set.

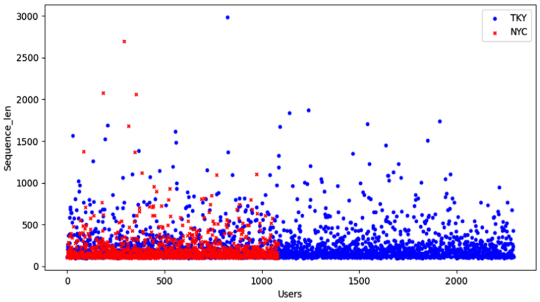

Fig. 4. The scatter chart of users

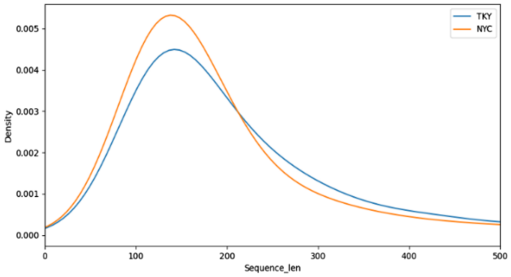

Fig. 5. The density chart of users

5.2 Baselines

To determine the effectiveness of our proposed method, we compare it with the following baselines:

Pop: It is a basic model that only recommends popular POIs.

BPR [23]: It uses the maximum posterior probability obtained by Bayesian analysis to rank and recommend POI.

GCMC [24]: It proposes a graph Auto-Encoder framework to solve the problem of rating prediction in the recommendation system from the perspective of link prediction.

SASRec [20]: It uses a left-to-right Transformer language model to capture users' sequential behaviors, and achieves sequential recommendation.

5.3 Metrics

To evaluate the performance of our proposed method, we employ precision and recall. Precision at a cutoff K, is denoted as *Pre*@*K*, and Recall at a cutoff *k*, is denoted as *Recall*@*K*. Where *K* is the number of predicted POIs in the result. These are general metrics for POI research used in previous work [25]. The *Pre*@*K* is the ratio of recovered POIs to the K predicted POIs, and *Rec*@*K* is the ratio of recovered POIs to the groundtruth. Given the user set U. We set the masked POIs as the groundtruth V_u^T,

and V_u^P is the set of prediction result. The definitions of $Pre@K$ and $Rec@K$ are shown as follows:

$$Pre@K = \frac{1}{|U|}\sum_{u\in U}\frac{\left|V_u^T \cap V_u^P\right|}{K} \tag{10}$$

$$Rec@K = \frac{1}{|U|}\sum_{u\in U}\frac{\left|V_u^T \cap V_u^P\right|}{\left|V_u^T\right|} \tag{11}$$

5.4 Settings

In our method, we use four heads of attention modules and two blocks of multi-head attention for the check-in sequence. We train our model using the Adam optimizer with a learning rate of 0.001 and set the dropout ratio to 0.1. The batch size is 16 and the dimension of location embedding is 256. The parameters of the baselines are the default values. The trend of the model's loss with the epoch during the training process is shown in the Fig. 6. The value of loss shows an oscillating decline with the increase of epoch. This is because the set batch size is relatively small, and the error difference of each batch training is large. When the epoch reaches 80, the loss of the model gradually begins to converge. Therefore, the number of training epochs is set to 100 for NYC and TKY.

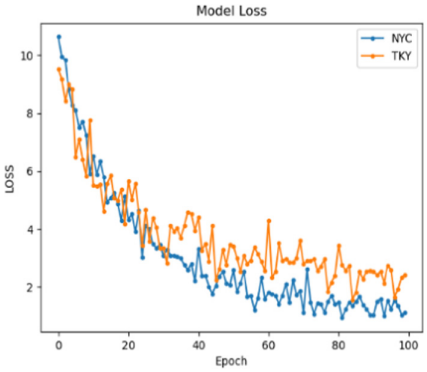

Fig. 6. The model loss with epoch

In the training process, the user behavior sequence are randomly masked to learn the relationship between POIs. In the testing process, in order to explore the influence of the continuity and discontinuity of the missing POIs in the sequence on the predictive ability of the model, we use mask in a fixed position in the user behavior sequence. We conduct multiple tests under both continuous and discontinuous conditions. The conditions are as follows (Table 2):

Table 2. Setting the number of masked POIs

The number of consecutively missing POIs	The total number of missing POIs
0	7
3	6
5	10
7	7

5.5 Comparison with Baselines

We set the parameters of the baselines to the default values. The number of missing POIs is 7 consecutively, and compare our method with baselines on the NYC and TKY datasets. For baselines, take the trajectory from the first POI to the missing POI in the dataset sequence. This is because baselines are all unidirectional models. In this way, we can compare the effect of our model using the information before and after the missing POIs. The comparison results are shown in the Figs. 7 and 8.

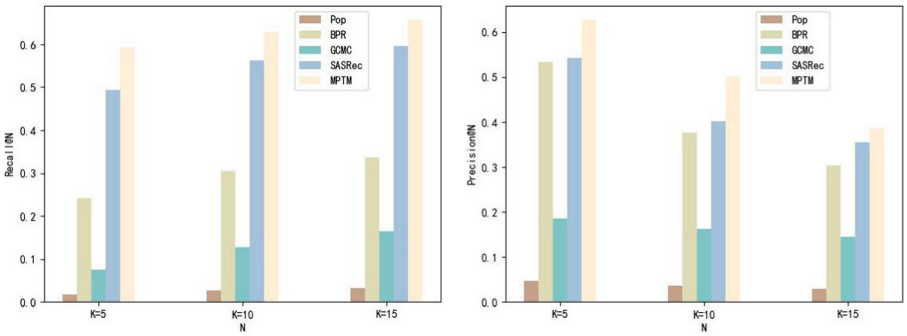

Fig. 7. Comparison with baselines on NYC

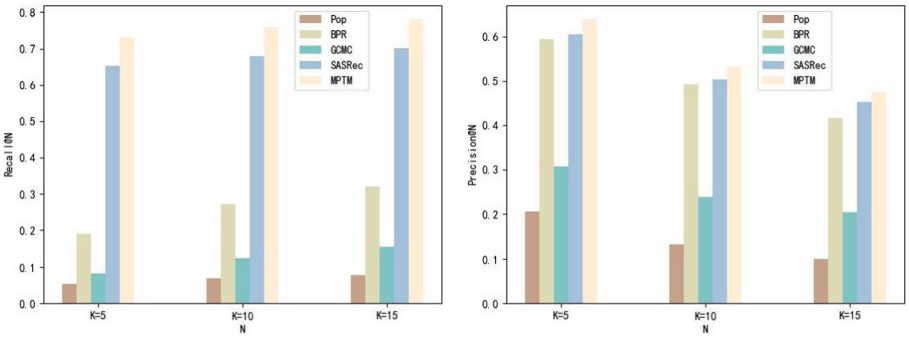

Fig. 8. Comparison with baselines on TKY

Our model MPTM is superior to other methods in recall and precision on both data sets. Pop only predicts based on the popularity of POI without considering the user's behavior habits, so it is much lower than other methods on recall and precision. GCMC constructs a bipartite graph of users and POI through the interaction between users and POI, but does not consider the user's behavior sequence, so the values on recall and precision are low. This also shows that the effect of constructing graph features alone for missing POI completion is not well, and further proves that the combination of graphic features and user behavior sequence is effective. The recall value of BPR on the two datasets is low, but the value of precision is close to SASRec and MPTM. This is because BPR is different from the general ranking model in that it reconstructs a partial order relationship for each user for personalized recommendation to predict. Both SASRec and MPTM use multi-head attention in transformer and consider the potential relationship between user behavior sequence and check-in POIs. But our model MPTM uses a bidirectional structure and takes into account the pre-order and post-order trajectory sequences with missing POIs.

5.6 Impact of POIs in Consecutive Masks

In order to explore the influence of the continuity and discontinuity of the missing POIs in the trajectory sequence on the predictive ability of the model, we use mask in a fixed position in the user behavior sequence. The experimental results are shown in the Tables 3 and 4, where number is the number of consecutive masked POIs. Table 3 is the experiment on the NYC, and Table 4 is the experiment on the TKY. Under all conditions, the recall of the two datasets is high, and the precision is relatively low. This shows that the model recalled many positively correlated POIs in the candidate set, but the number of correct hits is not high. There is not much difference between the recall of NYC and TKY under the four conditions. This shows that the continuity factor of the masked POI does not have much influence when the model recalls the relevant

Table 3. Experiments on NYC

Number	Recall@5	Recall@10	Recall@15	Pre@5	Pre@10	Pre@15
1	0.5974	0.6385	0.6693	0.5224	0.4467	0.3122
3	0.5985	0.6401	0.6657	0.4487	0.3838	0.2681
5	0.6109	0.6460	0.6713	**0.6107**	**0.4305**	**0.3355**
7	0.5916	0.6277	0.6574	0.5174	0.4391	0.3066

Table 4. Experiments on TKY

Number	Recall@5	Recall@10	Recall@15	Pre@5	Pre@10	Pre@15
1	0.7319	0.7643	0.7850	0.6404	0.5350	0.3663
3	0.7287	0.7611	0.7816	0.5465	0.4566	0.3126
5	0.7344	0.7638	0.7838	**0.7344**	**0.5092**	**0.3918**
7	0.7299	0.7599	0.7807	0.6387	0.5319	0.3643

positive samples. However, there is a gap between the precision on datasets, which fluctuates in the range of 0.3 to 0.7. This shows that the continuity factor of the masked POI has an impact on the model's accurate prediction of the missing POI.

In NYC and TKY, when the number of masked POIs is 5 consecutively, the values of recall and precision are higher. This shows that when the number of consecutive missing POIs is too small or too high, it will affect the predictive ability of the model. Especially when k = 5, the value of precision on two datasets is the highest.

6 Conclusions

According to the scene analysis, we propose the missing POI problem in the trajectory sequence. Incomplete user POI trajectory sequence has negative impact on subsequent tasks such as POI recommendation, location prediction, and human mobility. In order to overcome the difficulty of combining the pre-order and post-order trajectory sequences with missing POIs, we propose a bidirectional model MPTM based on the transform encoder. It is combined with graph features to mine the relationship between the missing POI and the known POI in the trajectory sequence and the characteristics of the user behavior sequence. The results demonstrate that our MPTM outperforms the state-of-the-art methods in terms of performance metrics recall and precision. For future work, we consider combining the explicit features of POI such as category and geographic location to solve the problem of user sequence trajectory splicing.

Acknowledgements. This research is sponsored by Natural Science Foundation of Chongqing, China (No. cstc2020jcyj-msxmX0900), the Fundamental Research Funds for the Central Universities (Project No. 2020CDJ-LHZZ-040), and National Natural Science Foundation of China (Grant No. 72074036, 62072060).

References

1. Lian, D., Wu, Y., Ge, Y., Xie, X., Chen, E.: Geography-aware sequential location recommendation. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 2009–2019 (2020)
2. Zhao, K., et al.: Discovering subsequence patterns for next POI recommendation. In: IJCAI, pp. 3216–3222 (2020)
3. Zhang, W., Wang, J.: Location and time aware social collaborative retrieval for new successive point-of-interest recommendation. In: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, pp. 1221–1230 (2015)
4. Zhang, L., Sun, Z., Zhang, J., Lei, Y., Klanner, F.: An interactive multi-task learning framework for next POI recommendation with uncertain check-ins. In: Twenty-Ninth International Joint Conference on Artificial Intelligence and Seventeenth Pacific Rim International Conference on Artificial Intelligence {IJCAI-PRICAI-20} (2020)
5. Zhao, S., Zhao, T., Yang, H., Lyu, M.R., King, I.: STELLAR: spatial-temporal latent ranking for successive point-of-interest recommendation. In: Thirtieth AAAI Conference on Artificial Intelligence (2016)

6. Liao, D., Liu, W., Zhong, Y., Li, J., Wang, G.: Predicting activity and location with multi-task context aware recurrent neural network. In: IJCAI, pp. 3435–3441 (2018)
7. Nawaz, A., Huang, Z., Wang, S., Akbar, A., AlSalman, H., Gumaei, A.J.S.: GPS trajectory completion using end-to-end bidirectional convolutional recurrent encoder-decoder architecture with attention mechanism. Sensors **20**, 5143 (2020)
8. Zheng, K., Zheng, Y., Xie, X., Zhou, X.: Reducing uncertainty of low-sampling-rate trajectories. In: 2012 IEEE 28th International Conference on Data Engineering, pp. 1144–1155. IEEE, (2012)
9. Liu, Q., Wu, S., Wang, L., Tan, T.: Predicting the next location: a recurrent model with spatial and temporal contexts. In: Thirtieth AAAI Conference on Artificial Intelligence (2016)
10. Koren, Y., Bell, R.: Advances in Collaborative Filtering, pp. 77–118. Springer, Boston, MA (2015)
11. Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., Jiang, P.: BERT4Rec: sequential recommendation with bidirectional encoder representations from transformer. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, pp. 1441–1450 (2019)
12. Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: Bert: pre-training of deep bidirectional transformers for language understanding (2018)
13. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)
14. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 701–710 (2014)
15. Han, P., Li, Z., Liu, Y., Zhao, P., Shang, S.: Contextualized point-of-interest recommendation. In: Twenty-Ninth International Joint Conference on Artificial Intelligence and Seventeenth Pacific Rim International Conference on Artificial Intelligence {IJCAI-PRICAI-20} (2020)
16. Zhao, P., Zhu, H., Liu, Y., Xu, J., Zhou, X.: Where to go next: a spatio-temporal gated network for next poi recommendation. Comput. Sci. **33**, 5877–5884 (2019)
17. Liu, X., Liu, Y., Li, X.: Exploring the context of locations for personalized location recommendations. In: IJCAI, pp. 1188–1194 (2016)
18. Lu, Y.S., Shih, W.Y., Gau, H.Y., Chung, K.C., Huang, J.L.: On successive point-of-interest recommendation. Anthology **22**, 1151–1173 (2019)
19. Manotumruksa, J., Macdonald, C., Ounis, I.: A contextual attention recurrent architecture for context-aware venue recommendation. In: The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 555–564 (2018)
20. Kang, W.-C., McAuley, J.: Self-attentive sequential recommendation. In: 2018 IEEE International Conference on Data Mining (ICDM), pp. 197–206. IEEE, (2018)
21. Wu, Y., et al.: Google's neural machine translation system: bridging the gap between human and machine translation (2016)
22. Yang, D., Zhang, D., Zheng, V.W., Yu, Z.: Modeling user activity preference by leveraging user spatial temporal characteristics in LBSNs. IEEE Trans. Syst. Man Cybern. Syst. **45**, 129–142 (2014)
23. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: Bayesian Personalized Ranking from Implicit Feedback (2012)
24. van den Berg, R., Kipf, T.N., Welling, M.: Graph convolutional matrix completion (2017)
25. Yu, F., Cui, L., Guo, W., Lu, X., Li, Q., Lu, H.: A category-aware deep model for successive poi recommendation on sparse check-in data. In: Proceedings of the Web Conference 2020, pp. 1264–1274 (2020)