Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 3698198, 10 pages
https://doi.org/10.1155/2018/3698198

Research Article

Personalized POI Recommendation Based on Subway Network Features and Users' Historical Behaviors

Danfeng Yan, Xuan Zhao, and Zhengkai Guo

State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China

Correspondence should be addressed to Zhengkai Guo; zhengk_guo@bupt.edu.cn

Received 16 April 2018; Accepted 14 June 2018; Published 9 July 2018

Academic Editor: Kok-Seng Wong

Current recommender systems often take fusion factors into consideration to realize personalize point-of-interest (POI) recommendation. Historical behavior records and location factors are two kinds of significant features in most of recommendation scenarios. However, existing approaches usually use the Euclidean distance directly without considering the traffic factors. Moreover, the timing characteristics of users' historical behaviors are not fully utilized. In this paper, we took the restaurant recommendation as an example and proposed a personalized POI recommender system integrating the user profile, restaurant characteristics, users' historical behavior features, and subway network features. Specifically, the subway network features such as the number of passing stations, waiting time, and transfer times are extracted and a recurrent neural network model is employed to model user behaviors. Experiments were conducted on a real-world dataset and results show that the proposed method significantly outperforms the baselines on two metrics.

1. Introduction

Recommender system has always been a hot topic and a lot of approaches have been proposed up to now. Nowadays, there are varieties of recommender systems in people's daily life, providing news, music, video, goods, or other kinds of items to the users. For example, Netflix is a famous online video rental provider, which recommends videos to its users based on historical rating data, and online retailers like Amazon and Alibaba provide products recommendation services on their websites and guide the customers' consumption actions. In recent years, with the development of mobile Internet, people get to use mobile APPs to find points of interest such as restaurants, malls, and view spots. In this situation, a good POI recommender system could show the most suitable candidate targets to users, thus helping to save users' time for picking and attain higher satisfaction.

Current recommender systems often use fusion factors, like users' profile, historical behaviors, and attributes of items, to calculate the relevance between users and items. Different from recommender systems for items like news or music, the location and other geographical factors are more important in personalized POI recommendation. This is because the distance between the points of interest and the user mainly determines the travel time, and people usually have more activities like eating, shopping, or watching movies in the region nearby. Some previous work has focused on applying location features in recommender systems: Natarajan and Moh [1] added a unique new feature of location preference to their news recommender system. Liu et al. [2] proposed a framework to recommend potential customers to vendors in location-based social network. Lee et al. [3] proposed a recommender system integrating location context, personal context, environment context, and user preference. However, these approaches only consider the geographical distance provided by location services such as GPS and use the location features from a single dimension. In fact, location-based recommendation problem is more complicated and more factors should be taken into consideration. In the real world, people cannot reach their destination in a straight line in most cases; thus the distance calculation should be based on road network information. In addition, people usually have to take the transport when they travel at a distance, and public transports such as bus or subway all have their own

lines and running schedules. In this context, factors such as station location, waiting time, and transfer times have varying degrees of impact on users' selection and should be taken into account in the POI recommender system. Compared with the bus system, the subway network is simpler and easier to analyze. At the same time, the subway is not susceptible to traffic jams, and the estimation of the time cost is more accurate. Therefore, our work focused on extracting the subway network features and using them to enhance the effect of POI recommendation.

Most recommender systems learn the users' preferences from their historical behaviors. For example, movie recommendations often predict users' preferences for new movies based on their rating scores on watched ones; the news recommender systems usually predict the click-through rate of news by analyzing users' actions like clicking, commenting, and sharing. In this context, how to represent and make use of the user preferences features has become a key issue. The user-based collaborative filtering algorithm represents user preferences in the form of vectors and uses them to calculate the similarity between users. The dimension of the vector is usually equal to the number of recommended items. The main disadvantage of this method is that the vector is very sparse in most cases and the similarity measure may not work. Matrix Factorization algorithms such as Probabilistic Matrix Factorization (PMF)[4], Nonnegative Matrix Factorization (NMF)[5] and Singular Value Decomposition (SVD) overcome this problem by using latent factors to represent user preferences. These methods consider the different aspects of user preferences equally but do not reflect the changes in users' interests. In fact, users' interests often change over time. In order to model the variation trend of user preferences over time, we trained a recurrent neural network using users' historical behaviors data and use the state of the units in the hidden layer to represent the user preferences. In this way, we took the user preferences as input features and added them into the ranking model of our POI recommender system.

In summary, the main contributions of our work include the following:

(1) We proposed a POI recommender system based on merged features. We took restaurant recommendation as the study case and managed to merge user profile, users' historical behavior, restaurant information, and subway network features.

(2) We represented the variation trend of user preferences over time as a fixed-length vector by employing a recurrent neural network.

(3) We used the Wide & Deep [6] model to predict the scores given to the restaurants by users.

(4) We conduct experiments on a real-world dataset to evaluate the effectiveness of our method. Experimental results show that our method outperforms 4 baseline methods in terms of root mean squared error (RMSE) and Mean Absolute Error (MAE).

The rest of this paper is organized as follows. Section 2 reviews the related works. Section 3 presents the main approach of the proposed restaurant recommender system. Section 4 introduces the conducted experiments and analyzes the results. Finally, Section 5 concludes the paper.

2. Related Work

In this section, we mainly review the related works from three aspects, including recommendation approaches, methods on processing users' historical behaviors data, and ranking models commonly used in recommender systems.

2.1. Recommendation Approaches. There are varieties of recommendation strategies, including collaborative filtering algorithm, content based recommendation, association rule based recommendation, and knowledge based recommendation. Collaborative filtering algorithm is one of the most commonly used methods in recommender systems. The main idea of collaborative filtering algorithms is to predict the unknown rating score based on the known ones. Collaborative filtering algorithm can be further divided into user-based collaborative filtering and item-based collaborative filtering [7]. The user-based collaborative filtering algorithm supposes that you may also like the things liked by the people who have the same preference as you. Therefore, the first step of the algorithm is to find users' nearest neighbors according to the distance metrics such as Pearson correlation coefficient and cosine similarity. The user-based collaborative filtering algorithm has a performance bottleneck that the complexity of finding the nearest neighbors increases substantially as the number of users growing. In this context, the item-based method turns to compute the similarity between different items, which reduces the computation cost and can be done offline. However, both of these two algorithms have the disadvantage that the user-item matrix could be so sparse that the distance metric may not work very well. To overcome this issue, Matrix Factorization algorithms factorize the user-item matrix and reduce the dimension of the parameters. The content based recommendation works based on users' historical behaviors, which is widely employed in information retrieval. Compared with the collaborative filtering methods, the content based recommendation models the attributes of the item and the results are easier to explain. The association rule based method makes the recommendation by mining the relevance between different items and has been successfully applied in retailing. The knowledge based recommendation utilizes the functional knowledge and makes the inference. Since the domain knowledge is hard to obtain, the knowledge based recommendation has greater limitations compared with other strategies.

Specifically, Wang et al. [8] unified user-based and item-based collaborative filtering approaches by similarity fusion and made the model more robust to data sparsity. Yao et al. [9] used a probabilistic generative model to take both the rating data and the semantic content data into consideration in web service recommendation. Tewari et al. [10] proposed a book recommendation system based on combine features of content based filtering, collaborative filtering, and association rule mining. Zhu et al. [11] take into account the importance of location and the intragroup influence in POI

group recommendation and employ distance prefiltering and distance ranking adjustment to improve recommendation satisfaction.

Quality-of-service (QoS) of restaurants is an important factor to be considered when recommending restaurants to users. Wang and Zheng et al. [12] propose a reputation measurement approach and a malicious feedback ration prevention scheme of web services. Wang and Ma et al. [13] present an integrated QoS prediction approach, which unifies the modeling of multidimensional QoS data via a multilinear-algebra based concept of tensor, for web service recommendations. Wang and Zhao et al. [14] propose a service recommendation approach based on collaborative filtering and make QoS prediction using user mobility.

In this paper, we recommend the restaurants to the user by predicting the user's rating score on each restaurant. In this way, the recommendation task is considered as a regression problem. Multidimensional features, such as user profile, restaurant attributes, users' historical behaviors, and subway network features, are selected as the input of the model. Compared with other methods, this approach is more scalable and has higher accuracy. The proposed architecture will be illustrated in detail in Section 3.

2.2. Processing History Data. Users' historical behaviors data are very significant in recommender systems. This kind of data, such as the list of videos that have been watched, purchase records, and browser history, reflects the users' preference to a certain extent and is of great significance for predicting users' future actions [15–18]. A simple way to make use of this kind of data is to concatenate users' historical behaviors in series and represent these records with a vector. This method is widely used in recommender systems of video websites, online shopping, and so on. However, it has some disadvantages and may have a bad impact on the prediction model. To be specific, users usually have different number of historical behavior records, as a result of this the length of the vector is not fixed and cannot be input into the prediction model directly. Some techniques are employed to solve this problem, such as truncation and completion. As shown in Figure 1, for the user who has few historical behavior records, the completion processing fills the vacancy position of the vector by zero. Correspondingly, for the user who has more historical actions, the truncation processing abandons the earlier records and just keeps the newer data. By these means the historical behaviors feature vector of different users is fixed-length and represented in a unified form. The truncation and completion processing generate the fixed-length vector; nevertheless, these operations bring the problems such as information loss and introduction of noise at the same time. Moreover, the dimension of the vector may be so high that influences the efficiency of model training.

Another common method of processing users' historical behavior records is employing the embedding technique. For example, the video recommender system of YouTube averages the embeddings of the sparse video IDs and the embeddings of the historical search tokens and then inputs the watch vector and the search vector to the hidden layers of the neural network model [19]. Zheng et al. [20] proposed

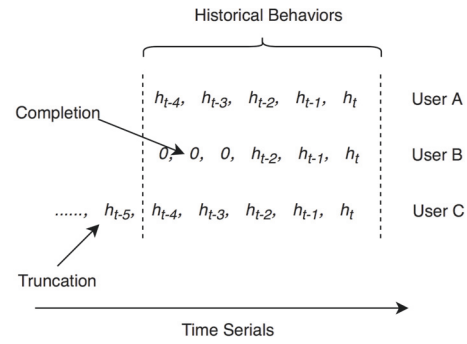

Figure 1: Example of historical behavior records processing method.

a model named DeepCoNN, which utilizes two parallel convolutional neural networks to model users' behaviors and the properties of items from the reviews and designs a shared layer to couple the two networks together. As illustrated in Figure 2, this method learns the latent factors of users' preferences and the properties of items in a similar way as the Matrix Factorization algorithms. Actually, the Matrix Factorization algorithms can also be considered as a sort of embedding technique, where the matrices after decomposition represent the embeddings of users and items. These methods compress the variable sized user behavior data into a fixed-length vector but do not make the most of the timing characteristics of the data. The user behavior sequence reflects the user behavior patterns and the variation trend of user preferences over time; thus, the sequence order should be taken into consideration when modeling. The recurrent neural network is suitable for modeling the time series data and has been successfully applied in processing history records in recommender systems. Specifically, Liu et al. [21] proposed an extension model of recurrent neural network which outputs the embeddings of the sequential heterogeneous attributes for item recommendation. Dai et al. [22] put forward a point of view that user features and item features may drift, evolve, and coevolve over time due to the interaction, and they employed a recurrent neural network to learn the representation of influences automatically and capture the coevolving nature of both user feature and item feature. In the news recommendation method introduced in [23], a recurrent neural network is used to generate the representation of the user with browsing histories as the input.

In the proposed recommender system, we use a single layer recurrent neural network to learn the behavior pattern and the preference of the user. In the training phase, users' historical behavior data are organized in sequence and fed into the model with a sliding window. After the model is well trained, we input the historical record of the user into the model and use the final states of the hidden layer as the vector representation of the user behavior feature.

2.3. Models. Varieties of machine learning models have been employed for ranking in recommender systems. Logistic Regression, which is one of the most famous and commonly

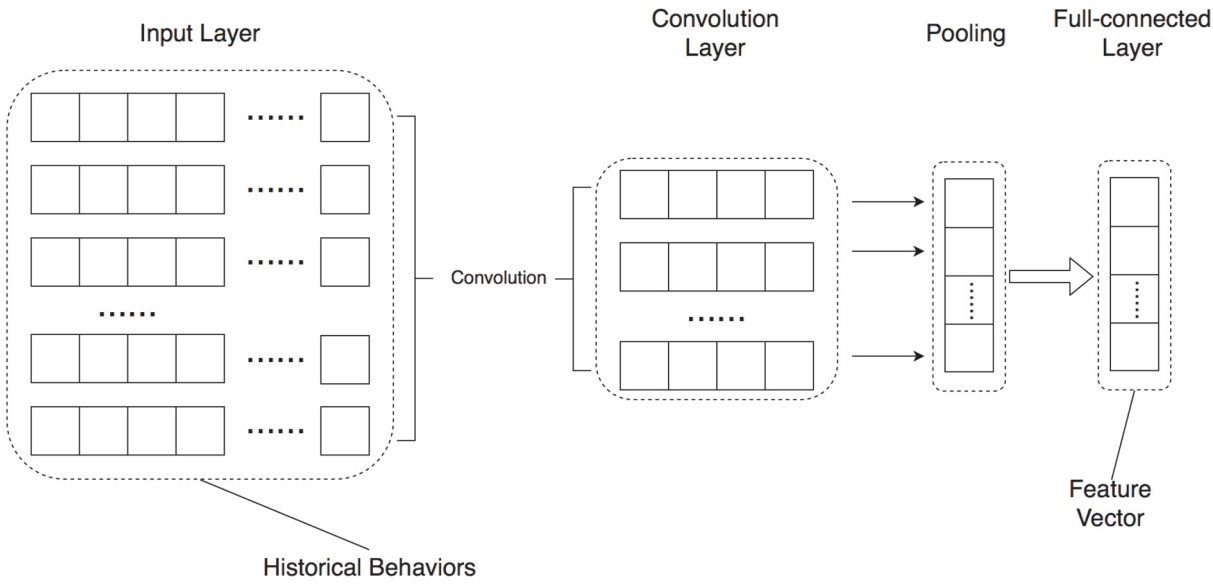

Figure 2: Encode historical behaviors with convolutional neural network.

used algorithm, has been proved to be effective in application scenarios such as online advertising, recommender systems, and web search engine. The main advantages of Logistic Regression algorithm are easy implementation, high computing efficiency, easy to parallelize, and suitable for processing high dimensional and sparse features. As a generalized linear model, Logistic Regression algorithm weights and sums the feature values and maps the result to the interval (0, 1) with the logistic function. Therefore, the algorithm requires much manual feature engineering to filter and validate the combined features, which is very complex and lacks efficiency. In this context, Rendle proposed an advanced model named Factorization Machines (FMs) [24]. Factorization Machines add the explicit combined features to the equation and use factorized parameters to estimate the interactions between variables. Field-aware Factorization Machines (FFMs) [25] are variant of FMs, and the model is based on the idea that features can be grouped into fields for most datasets. Every variable in FMs has only one parameter vector representing the latent effect with other variables, but in FFMs, each variable has several latent vectors to interact with variables belonging to different fields. Neural networks have a strong nonlinear approximation ability and have higher prediction accuracy in most cases. With the development of deep learning, there have been several approaches to employ deep neural network to process the ranking task in recommender systems. Among these approaches, the Wide & Deep model proposed in [6] joins the generalized linear models and the deep neural networks and trains the two components synchronously. In this model, the generalized linear model (wide component) is designed to learn the memorization of feature interactions, and the deep neural network (deep component) can learn the dense embeddings and generalize to feature combinations. Since the Wide & Deep model combines the benefits of memorization and generalization for recommender systems, we select this model to make the prediction in this paper.

3. System Design

In this section, we introduce the system design of the proposed restaurant recommender system from four main aspects. The first part presents the overall architecture of the recommender system and the major functions of all the components. Then, the extraction of subway network features is introduced in the second part. The third part illustrates the user behaviors modeling method. Finally, the ranking model of the recommender system is introduced in the fourth part of this section.

3.1. Architecture. The overall architecture of the proposed restaurant recommender system is illustrated in Figure 3.

As the figure shows, the recommender system consists of two main components, implementing the function of ranking model generation and restaurant recommendation, respectively. The ranking model generation component can further be divided into three parts: subway network features extraction, user behaviors modeling, and ranking model training. Concretely, the subway network feature extraction part firstly attaches the user and candidate restaurants to the nearest subway stations and then calculate the features such as distance from the station, number of stations passing through, and transfer times based on the subway network structure. The user behaviors modeling part learns user preferences and behavior patterns through the historical visiting records and generates users' behavior feature vectors. Finally, these features are organized as the input and employed to train the ranking model of the recommender system. As for the restaurant recommendation component, the system firstly concatenates users' basic information, candidate restaurants

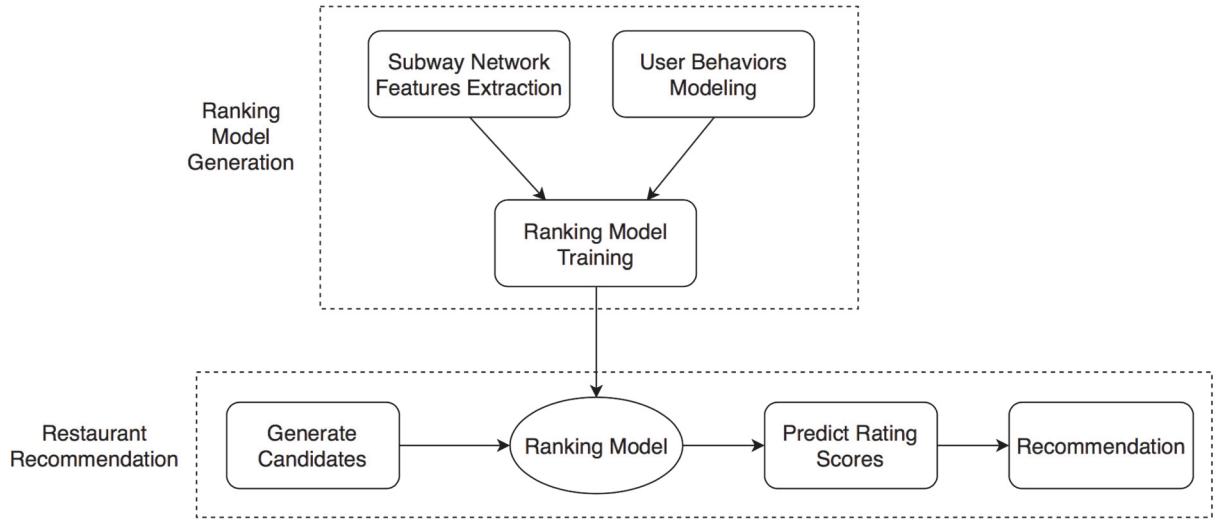

Figure 3: The overall architecture of the proposed restaurant recommender system.

characteristics, subway network features, and user behavior features together to obtain the input feature vectors. Then, the well trained ranking model is used to predict users' rating scores on each candidate restaurant. Finally, the system will make the recommendation based on the prediction results.

The subway network features extraction, user behaviors modeling method, and ranking model design are the focus of our work. These three aspects will be introduced in detail in the following subsections.

3.2. Subway Network Feature Extraction. Buses and subways are two of the most important ways of public transportation. In this paper, we select the subway network as the study object because it has some advantages comparing with buses. Firstly, the subway network is more popular with users when they travel by public transport. Secondly, the subway has a more accurate timetable and is not susceptible to traffic jams, which makes it more accurate to estimate the time cost. Thirdly, users prefer to take the subway when the destination is far away. For above-mentioned reasons, we focus on analyzing the subway network and extract a series of features. Specifically, we studied the subway network of Guangzhou City, which includes 13 lines and 205 stations up to now.

The main steps of subway network features extraction are as follows:

(1) The first step of subway network features extraction is finding the nearest stations to the user and the candidate restaurant. As a nearest neighbor problem, one possible solution is to calculate the Euclidean distance between the object and every subway stations. However, this method is so inefficient that is unable to meet the needs of real-time recommendation. In our work, we constructed a KD-Tree [26] based on the latitude and longitude of the subway stations. By this means, the city map is partitioned into areas and the lookup complexity is reduced from $O(n)$ to $O(\sqrt{n})$, where n represents the number of subway stations.

(2) When the first step is finished, the nearest stations to the user and candidate restaurant are set as the origin and the destination respectively and then calculate the number of stations passing through. This is a typical shortest path problem and we employ the Floyd algorithm [27] to calculate the shortest path between any two stations.

(3) The number of stations passing through reflects the distance between two subway stations. However, the distance between adjacent stations is often different, and it is inaccuracy and insufficient to use only this feature. Moreover, the travel time is the most intuitive feeling for users. Thus, we additionally calculate the travel time after getting the shortest path. The travel time is calculated according to the subway schedule, which can be obtained through Internet.

These subway network features improve the location-based recommendation approach by taking into consideration the transportation conditions. Moreover, measuring the distance based on the number of stations passing through and the travel time is more appropriate than calculating the Euclidean distance on the plane directly.

3.3. Modeling User Behaviors. As discussed in Section 2, we use a three layers RNN to learn the behavior pattern and the preference of users. The model structure and the main process of the method are shown in Figure 4.

The modeling process consists of two stages: model training and feature vector calculation. Firstly, we constructed the feature vector of every restaurant. There are four parts in the feature vector: restaurant ID, restaurant type, average consumption, and average rating scores. As categorical features, the ID and type of the restaurant are transformed to vector representation by embedding method. Moreover,

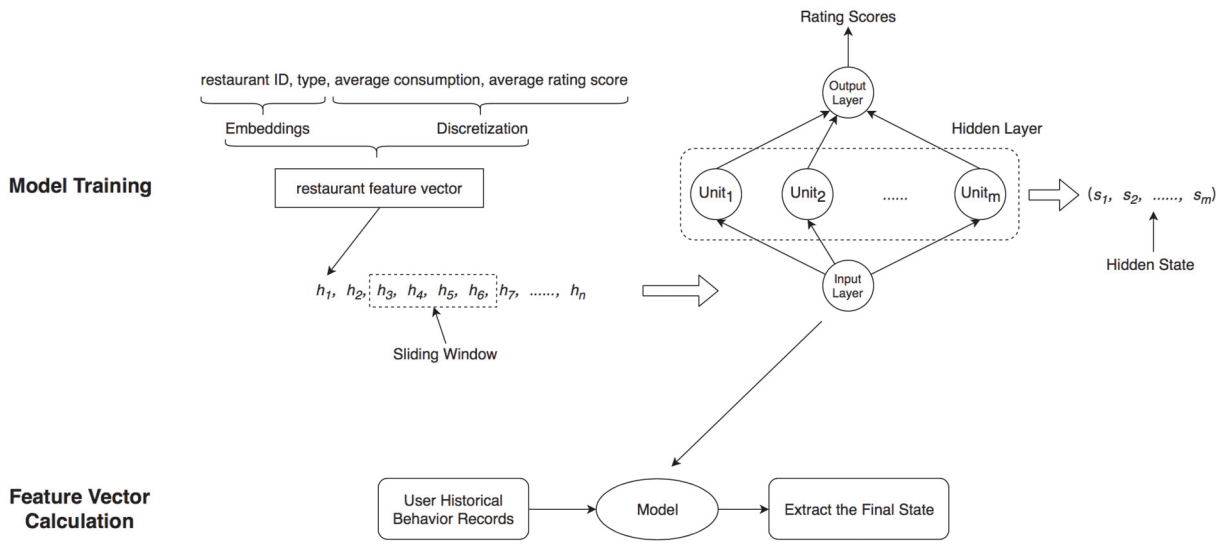

Figure 4: The RNN based user behaviors modeling method.

the other two continuous variables are transformed to the vector form by using equal-depth discretization. The neural network model consists of three layers, including input layer, output layer, and hidden layer. The hidden layer contains m units where m determines the dimension of users' preference feature vector. In the training phase, for each user, the feature vectors of the visited restaurants are arranged in order and fed into the model with a sliding window. User's rating score on the next restaurant behind the sliding window is set as the prediction target and the optimization target is square loss. According to the principle of the recurrent neural network, the hidden state is updated based on both the current input and the hidden state of the last time-step. In this context, we can conclude that the final hidden state contains the whole context information of the input users' behaviors. Furthermore, the final hidden state reflects users' preferences and behavior patterns, and it is reasonable to be used as the behavior feature vector of users.

After the recurrent neural network is well trained, it is saved and employed to generate the behavior feature vector of the user. Assuming that the sliding window length is T, we fed into RNN model the user's last T visited restaurants and extract the final state of hidden layer. We regard the final state of hidden layer as user's behavior feature vector. In this approach, the RNN model handles the varieties in user's behaviors over time.

There are mainly two key hyperparameters in this user behaviors modeling method: the units number in the hidden layer and the size of the sliding window. Both of these two hyperparameters have a great impact on the modeling effect. Specifically, the feature vector can hardly express users' preferences if there are few units in the hidden layer. By contrast, too many units generate high dimensional feature vector, which is usually very sparse and fails to compress users' historical behavior data. For the size of the sliding window or the number of time-steps, the model needs

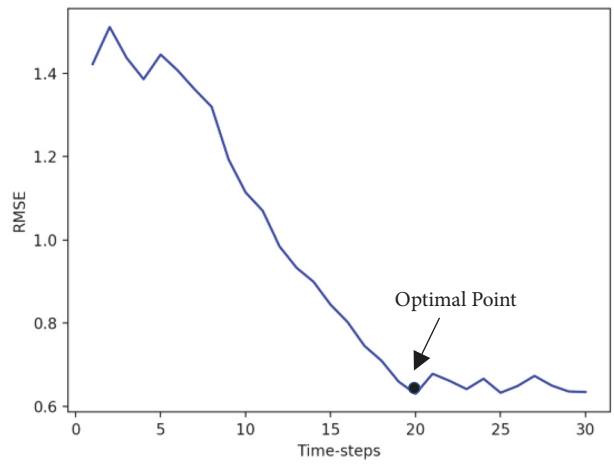

Figure 5: Prediction deviation with various time-steps.

enough records to learn user characteristics. Theoretically, more historical records contain more information and can improve the modeling effect to a certain extent. However, for ordinary RNN model, later inputs usually have more weight than previous ones. Therefore, the proposed model pays more attention to users' recent behaviors and the long-term records only have a little influence. Actually, we tested the modeling effect with various sliding window sizes and found that the prediction deviation is approximately asymptotically convergent. The experiment result is shown in Figure 5; it shows that the sliding window size value has an "optimal point". When the size value is in the left interval of the point, the modeling effect has a significant improvement with historical records growing. Correspondingly, the modeling effect is tending to be stable for more input records exceeding the point. As to the issue of how to determine the optimal point, we have not come up with effective methods besides

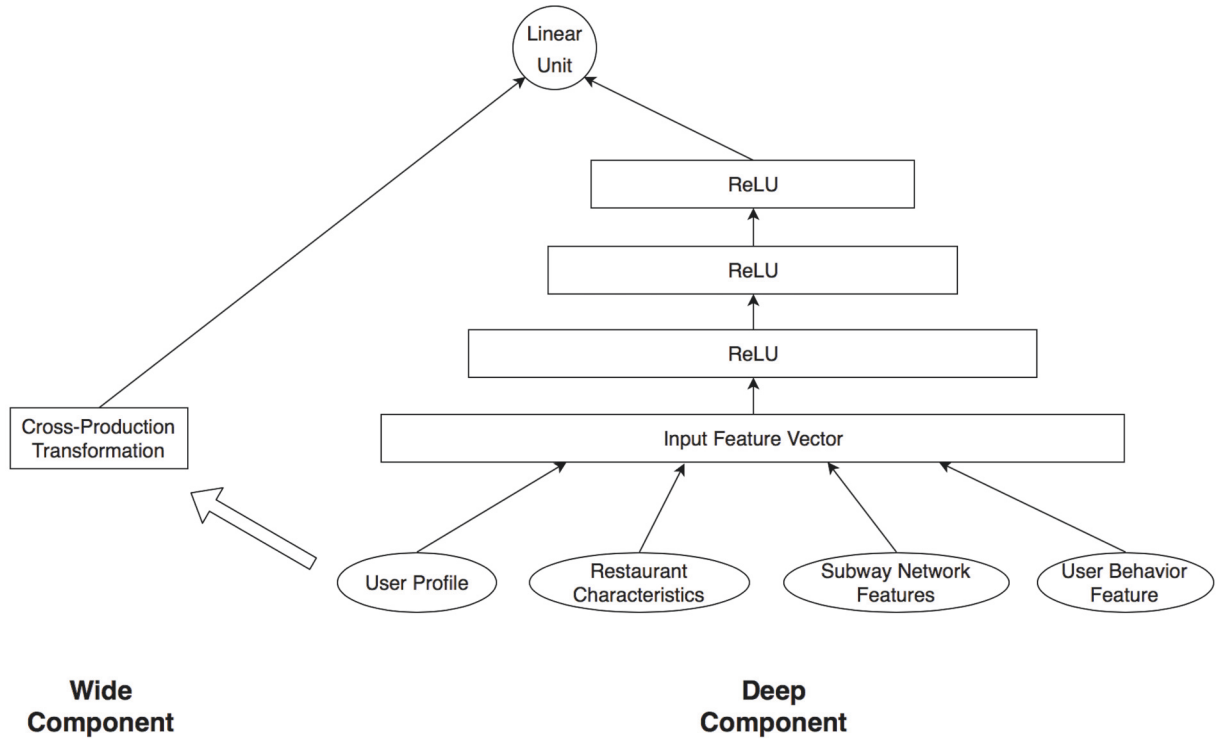

Figure 6: The Wide & Deep model structure designed for the ranking task of restaurant recommender system.

linear detection at present. We will focus on this question in our future work.

3.4. Ranking Model. The ranking model is always the core component of recommender systems. There are several training strategies for the task of learning to rank, including Pointwise, Pairwise, and Listwise. In this paper, we take the Pointwise method and solve the ranking task as a regression problem. Four types of features, including users' basic information, candidate restaurants characteristics, subway network features, and user behavior features, are concatenated as the input of the model. Users' rating scores on restaurants are set as the learning target and square error is set as the loss function. As mentioned in Section 2.3, we employed the Wide & Deep model to predict the scores, and the model structure is shown in Figure 6.

The model consists of two parts: the wide component made up of a generalized linear model and the deep component of a deep neural network. According to [6], the wide component is designed to learn the memorization of feature interactions and the deep component achieves the generalization. In our model, the cross-production transformation of binary features and the user historical behavior features are fed into the wide part. For the deep part of the model, the categorical features such as the user age and the type of the restaurant are transformed into embeddings firstly. Then all the embeddings and dense features are concatenated together and fed into the network which contains three ReLU layers. Finally, there is a linear unit output of the regression prediction result. The prediction result is shown as follows:

$$\text{prediction} = W_{wide}^{T} X_{wide} + W_{deep}^{T} a^{(l)} + \text{b} \tag{1}$$

where X_{wide} is the features fed into the wide part, $a^{(l)}$ is the activation of the final ReLU layer, W_{wide} and W_{deep} are the vectors of wide component weights and weights applied on $a^{(l)}$, respectively, and b is the bias term.

The Wide & Deep model is trained jointly. The gradients of the loss function is backpropagated from the output to both the wide and deep component of the model at the same time. Specifically, the wide part of the model is optimized using Adam method.

4. Experiments

In this section, we conduct extensive experiments with a real-word dataset. First, we present the dataset and the evaluation metric used in our experiments. Then, we describe the baseline algorithms selected for comparisons. Then, we introduce the parameter settings. Finally, we analyze the results.

4.1. Dataset and Evaluation Metric. In our experiments, we use a dataset collected from a real APP named Green Travel. It contains a large number of users, restaurants, and ratings for restaurant. We choose 1M ratings from 1525 users to 2657 restaurants. Each rating is an integer between 1 and 5.

Table 1: MAE and RMSE values of baselines and our model.

Model	MAE	RMSE
PMF	0.6142	0.8176
NMF	0.5863	0.7207
SVD	0.8218	1.0344
SVD++	0.8221	0.9352
W&D	0.5532	0.6738
W&D-his	0.5205	0.6419
W&D-sub	0.5398	0.6638
W&D-his-sub	**0.5148**	**0.6317**

In this paper, we use the most popular metrics, Root Mean Square Error (RMSE), and Mean Average Precision (MAE):

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (r_i - \hat{r}_i)^2}{n}} \quad (2)$$

$$MAE = \frac{\sum_{i=1}^{n} |r_i - \hat{r}_i|}{n} \quad (3)$$

where r_i denotes the ith observed rating, $\hat{r}_i$ denotes the ith predicted rating, and n is the number of ratings in test set.

4.2. Baselines. In order to evaluate the performance of our model, we compare it with the following state-of-the-art models.

(i) **NMF**: **N**onnegative **M**atrix **F**actorization. It only uses the rating matrix as the input.

(ii) **PMF**: **P**robabilistic **M**atrix **F**actorization. It models latent factors of users and items by Gaussian distributions.

(iii) **SVD**: **S**ingular **V**alue **D**ecomposition. It uses rating matrix as input and estimates two low-rank matrices. It also uses bias to reduce the error.

(iv) **SVD++**. It extends **S**ingular **V**alue **D**ecomposition by considering implicit feedback information for latent factor modeling [28].

4.3. Parameter Settings. We divided the dataset into three subsets: 80% for training set, 10% for validation set, and 10% for test set. All the hyperparameters of the baselines and our model are tuned with the validation set.

We used grid search to find the best value for the number of latent factors and regularization parameters. For NMF, PMF, SVD, and SVD++, we set the number of latent factors k = 64, 64, 48, 32, and regularization parameters r = 0.1, 0.1, 0.15, 0.2.

For our model, we use 1-layer RNN to model user's historical behaviors. Specifically, we used CuDNNGRU in order to benefit from its performance. The dimension of the user's latent factor is set to 128. We tried different number of user's historical behaviors and find 20 is the best. The number of hidden layers of the deep component in the model is 3.

All the neural matrix parameters in hidden layers and RNN layers are initialized from a uniform distribution between [-0.1, 0.1]. The network was trained via Adam optimizer with batch size 64 and learning rate 0.001.

Our models are implemented in Tensorflow, a well-known Python library for deep learning. Our models are trained and tested on an NVIDIA GTX1080 TI GPU.

4.4. Results and Discussion. We tried different feature combinations in our model:

(i) **W&D**: Wide & Deep model only use ratings, user profile, and restaurant information.

(ii) **W&D-his**: Wide & Deep model use ratings, user profile, restaurant information, and user's historical behaviors.

(iii) **W&D-sub**: Wide & Deep model use ratings, user profile, restaurant information, and subway network feature.

(iv) **W&D-his-sub**: Wide & Deep model use ratings, user profile, restaurant information, user's historical behaviors, and subway network feature.

Table 1 shows the best MAE and RMSE of PMF, NMF, SVD, SVD++, and our model with different features. We can observe from Table 1 that our models achieve better performance than all the baselines. It significantly outperforms the four other baselines in MAE and RMSE, which demonstrates the effectiveness of our models.

Moreover, W&D-his and W&D-sub outperform W&D, which shows that user historical behaviors and subway network features are important for restaurant recommendation. Furthermore, we can see that W&D-his obtains lower MAE and RMSE than W&D-sub, from Table 1, which validates user's historical behaviors, and is more effective than subway network feature. Finally, we add user's historical behaviors and subway network feature into our model, which obtains the best performance.

5. Conclusions

In this paper, we take the restaurant recommendation as an example and propose a POI recommender system based on fusion features. Subway network features, including the number of passing stations, waiting time, and transfer times, are extracted to measure the distance between the user and the restaurant in a more reasonable way than calculating the

Euclidean distance directly. We employ a recurrent neural network with one hidden layer to learn the embeddings of users' historical behaviors. In this way, the time characteristics of behaviors data are captured and user preferences can be represented with a fixed-length vector. Experiment results show that both the subway network features and user historical behavior features are helpful for restaurant recommendation and can improve the prediction accuracy of the ranking model.

Data Availability

The authors use a dataset collected from a real APP named Green Travel. It contains a large number of users, restaurants, and ratings for restaurant.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This paper is supported by "National 863 Project (no. 2015AA050204)", "State Grid Science and Technology Project (no. 520626170011)", and "Natural Science Foundation of China (no. 61532006 and no. 61320106006)".

References

[1] S. Natarajan and M. Moh, "Recommending news based on hybrid user profile, popularity, trends, and location," in *Proceedings of the 2016 International Conference on Collaboration Technologies and Systems, CTS 2016*, pp. 204–211, usa, November 2016.

[2] Y. Liu, P. Zhao, V. S. Sheng et al., "RPCV: Recommend Potential Customers to Vendors in Location-Based Social Network," in *Web-Age Information Management*, vol. 9098 of *Lecture Notes in Computer Science*, pp. 272–284, Springer International Publishing, Cham, 2015.

[3] B. Lee, H. Kim, J. Jung, and G. Jo, "Location-Based Service with Context Data for a Restaurant Recommendation," in *Database and Expert Systems Applications*, vol. 4080 of *Lecture Notes in Computer Science*, pp. 430–438, Springer, Berlin, Heidelberg, 2006.

[4] A. Mnih and R. Salakhutdinov R, "Probabilistic matrix factorization," *Advances in Neural Information Processing Systems*, pp. 1257–1264, 2008.

[5] D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," *Advances in neural information processing systems*, pp. 556–562, 2001.

[6] H. Cheng T, L. Koc, and J. Harmsen, *Wide & Deep Learning for Recommender Systems*, 2016.

[7] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: item-to-item collaborative filtering," *IEEE Internet Computing*, vol. 7, no. 1, pp. 76–80, 2003.

[8] J. Wang, A. P. De Vries, and M. J. T. Reinders, "Unifying user-based and item-based collaborative filtering approaches by similarity fusion," in *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 501–508, ACM, August 2006.

[9] L. Yao, Q. Z. Sheng, A. H. H. Ngu, J. Yu, and A. Segev, "Unified collaborative and content-based web service recommendation," *IEEE Transactions on Services Computing*, vol. 8, no. 3, pp. 453–466, 2015.

[10] A. S. Tewari, A. Kumar, and A. G. Barman, "Book recommendation system based on combine features of content based filtering, collaborative filtering and association rule mining," in *Proceedings of the 2014 4th IEEE International Advance Computing Conference, IACC 2014*, pp. 500–503, ind, February 2014.

[11] Q. Zhu, S. Wang, B. Cheng, Q. Sun, F. Yang, and R. N. Chang, "Context-Aware Group Recommendation for Point-of-Interests," *IEEE Access*, vol. 6, pp. 12129–12144, 2018.

[12] S. Wang, Z. Zheng, Z. Wu, M. R. Lyu, and F. Yang, "Reputation measurement and malicious feedback rating prevention in web service recommendation systems," *IEEE Transactions on Services Computing*, vol. 8, no. 5, pp. 755–767, 2015.

[13] S. Wang, Y. Ma, B. Cheng, F. Yang, and R. Chang, "Multi-dimensional QoS prediction for service recommendations," *IEEE Transaction on Services Computing*, 2016.

[14] S. Wang, Y. Zhao, L. Huang, J. Xu, and C.-H. Hsu, "QoS prediction for service recommendations in mobile edge computing," *Journal of Parallel and Distributed Computing*, 2017.

[15] J. Davidson, B. Liebald, J. Liu, P. Nandy, and T. Van Vleet, "The YouTube video recommendation system," in *Proceedings of the 4th ACM Recommender Systems Conference (RecSys '10)*, pp. 293–296, Barcelona, Spain, September 2010.

[16] X. Zhao, H. Luan, J. Cai et al., "Personalized video recommendation based on viewing history with the study on YouTube," in *Proceedings of the International Conference on Internet Multimedia Computing and Service*, pp. 161–165, ACM, 2012.

[17] S. Seko, M. Motegi, T. Yagi, and S. Muto, "Video content recommendation for group based on viewing history and viewer preference," in *Proceedings of the 2011 IEEE International Conference on Consumer Electronics, ICCE 2011*, pp. 351-352, usa, January 2011.

[18] A. V. Bodapati, "Recommendation systems with purchase data," *Journal of Marketing Research*, vol. 45, no. 1, pp. 77–93, 2008.

[19] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in *Proceedings of the 10th ACM Conference on Recommender Systems, RecSys 2016*, pp. 191–198, September 2016.

[20] L. Zheng, V. Noroozi, and P. S. Yu, "Joint Deep Modeling of Users and Items Using Reviews for Recommendation," in *Proceedings of the the Tenth ACM International Conference*, pp. 425–434, Cambridge, United Kingdom, Feburary 2017.

[21] K. Liu, X. Shi, and P. Natarajan, "Sequential Heterogeneous Attribute Embedding for Item Recommendation," in *Proceedings of the 2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 773–780, New Orleans, LA, November 2017.

[22] H. Dai, Y. Wang, R. Trivedi, and L. Song, "Recurrent Coevolutionary Latent Feature Processes for Continuous-Time Recommendation," in *Proceedings of the the 1st Workshop*, pp. 29–34, Boston, MA, USA, September 2016.

[23] S. Okura, Y. Tagami, S. Ono, and A. Tajima, "Embedding-based news recommendation for millions of users," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2017*, pp. 1933–1942, August 2017.

[24] S. Rendle, "Factorization machines," in *Proceedings of the 10th IEEE International Conference on Data Mining, ICDM 2010*, pp. 995–1000, Australia, December 2010.

[25] Y. Juan, Y. Zhuang, W.-S. Chin, and C.-J. Lin, "Field-aware factorization machines for CTR prediction," in *Proceedings of the 10th ACM Conference on Recommender Systems, RecSys 2016*, pp. 43–50, usa, September 2016.

[26] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[27] R. W. Floyd, "Algorithm 97: shortest path," *Communications of the ACM*, vol. 5, no. 6, p. 345, 1962.

[28] Y. Koren, "Factorization meets the neighborhood: a multi-faceted collaborative filtering model," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '08)*, pp. 426–434, New York, NY, USA, August 2008.