PDPNN: Modeling User Personal Dynamic Preference for Next Point-of-Interest Recommendation

Jinwen Zhong[1,2], Can Ma[1(✉)], Jiang Zhou[1], and Weiping Wang[1]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{zhongjinwen,macan,zhoujiang,wangweiping}@iie.ac.cn
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

Abstract. Next Point of Interest (POI) recommendation is an important aspect of information feeds for Location Based Social Networks (LSBNs). The boom in LSBN platforms such as Foursquare, Twitter, and Yelp has motivated a considerable amount of research focused on POI recommendations within the last decade. Inspired by the success of deep neural networks in many fields, researchers are increasingly interested in using neural networks such as Recurrent Neural Network (RNN) to make POI recommendation. Compared to traditional methods like Factorizing Personalized Markov Chain (FPMC) and Tensor Factorization (TF), neural network methods show great improvement in general sequences prediction. However, the user's personal preference, which is crucial for personalized POI recommendation, is not addressed well in existing works. Moreover, the user's personal preference is dynamic rather than static, which can guide predictions in different temporal and spatial contexts. To this end, we propose a new deep neural network model called Personal Dynamic Preference Neural Network(PDPNN). The core of the PDPNN model includes two parts: one part learns the user's personal long-term preferences from the historical trajectories, and the other part learns the user's short-term preferences from the current trajectory. By introducing a similarity function that evaluates the similarity between spatiotemporal contexts of user's current trajectory and historical trajectories, PDPNN learns the user's personal dynamic preference from user's long-term and short-term preferences. We conducted experiments on three real-world datasets, and the results show that our model outperforms current well-known methods.

Keywords: Point-of-interest recommendation · User personal dynamic preference · Recurrent Neural Network

1 Introduction

Due to the prevalence of smart mobile devices, people frequently use Location-Based Social Networks (LBSNs) such as Foursquare, Twitter, and Yelp to post

V. V. Krzhizhanovskaya et al. (Eds.): ICCS 2020, LNCS 12142, pp. 45–57, 2020.
https://doi.org/10.1007/978-3-030-50433-5_4

check-ins and share their life experiences. Point-of-interest (POI) recommendation has become an important way to help people discover attractive and interesting venues based on their historical preferences. Many merchants also use POI recommendation as an important channel to promote their products.

POI recommendation has been extensively studied in recent years. In the literature, Markov Chains (MC) [1,2], collaborative filtering [3–6], and Recurrent Neural Networks [7–11] are three main approaches. These methods commonly focus on capturing either the user's short-term preferences or long-term preferences. User's short-term and long-term preferences are both important for achieving higher accuracy of recommendation [12]. The short-term preference is greatly influenced by time and POIs the user has visited recently, while the long-term preference reflects the user's personal habits thus all historically visited POIs need to be considered. However, the user's personal long-term preference, which is crucial for personalized POI recommendation, is not addressed well in existing work. Moreover, the user's personal preference is dynamic rather than static, which can guide predictions in different temporal and spatial contexts. To this end, we propose a new deep neural network model called Personal Dynamic Preference Neural Network (PDPNN). The core of the PDPNN model includes two parts: one part learns the user's personal long-term preferences from the historical trajectories, and the other part learns the user's short-term preferences from the current trajectory. By introducing a similarity function that evaluates the similarity between spatiotemporal contexts of user's current trajectory and historical trajectories, PDPNN learns the user's personal dynamic preference from user's long-term and short-term preferences.

We summarize our contributions in this paper as follows:

- We propose a new approach to better model dynamic user preference for their next POI from their current trajectory in conjunction with their trajectory context. Based on the attention-enhanced Long Short-Term Memory (LSTM) neural network model, we build a new neural network model named PDPNN for next POI recommendation.
- To accelerate training of trajectory contexts, we propose inner epoch cache mechanism to store the trajectory context output states during the process of model training. This reduces the complexity of the user's personal dynamic preference module from $O(n^2)$ to $O(n)$ in each training epoch.
- We conducted experiments on three real world datasets and the results show that our model outperforms current well known methods.

2 Related Work

Plenty of approaches have been proposed that focus on sequential data analysis and recommendation. Collaborative filtering based models, such as Matrix Factorization (MF) [3] and Tensor Factorization(TF) [4] are widely used for recommendation. These methods aim to cope with data sparsity and the cold start problem by capturing common short-term preferences among users, but cannot capture the user's personal long-term preferences. Other existing studies employ

the properties of a Markov chain to capture sequential patterns, such as Markov Chain (MC) [1] and Factorizing Personalized Markov Chains (FPMC) [2]. FPMC models user's preference and sequential information jointly by combining factorization method and Markov Chains for next-basket recommendation. However, both MC and FPMC methods fall short in learning the long-term preference and the periodicity of the user's movement.

Recently, as a result of the success of deep learning in speech recognition, vision and natural language processing, Recurrent Neural Networks (RNNs) have been widely used in sequential item recommendation [7–11]. Spatial Temporal Recurrent Neural Networks (STRNN) [8] utilizes a RNN architecture and linear interpolation to learn the regularity of sequential POI transition. However, traditional RNNs suffer from the issues of vanishing gradients and error propagation when they learn long-term dependencies [13]. Therefore, special gating mechanisms such as Long Short-Term Memory network (LSTM) [14] have been developed and widely used in recent work [9–11]. By controlling access to memory cells, LSTMs can alleviate the problem of long-term dependencies.

Attention mechanism is a key advancement in deep learning in recent years, and it shows a promising performance improvement for RNNs [15–17]. By introducing the attention mechanism, Attention-based Spatio-Temporal LSTM network (ATST-LSTM) [9] can focus on the relevant historical check-in records in a check-in sequence selectively using the spatiotemporal contextual information. However, these models are designed to learn short-term preferences and are not well suited to learning personal long-term preferences. They commonly add user embedding to RNN outputs, and reduce the problem of user context learning to the problem of learning a static optimal user embedding representation. It is more difficult to learn long-term preferences with simple attention mechanisms, especially for users with only a small set of historical trajectories.

3 Model Description

3.1 Problem Formulation

Let $U = \{u_1, \ldots, u_m\}$ denote the user set and $P = \{p_1, \ldots, p_j\}$ denote the POI set, where $\|U\|$ and $\|P\|$ are the total numbers of users and POIs, respectively. Each POI p_k is associated with a geographic location $l_k = (l_a, l_o)$, where l_a and l_o denote the latitude and longitude of the POI location. For a user $u \in U$, a check-in behavior means u_i visits a POI p_k at time t_k, which is denoted as tuple (u, p_k, t_k). Any check-in sequence with all the time intervals of successive check-ins less than a threshold value T_{delta}, is called a trajectory. Obviously, a user's historical check-ins will be segmented into many trajectories. We denote $T_i^u = \{(u, p_1, t_1), \ldots, (u, p_n, t_n)\}$ as the i-th trajectory of the user u, with $t_k - t_{k-1} \leq T_{delta}$ for neighbor check-ins of the trajectory, and $\|T_i^u\|$ is the total number of check-ins. All the historical trajectories of the user u are denoted as $T^u = \{T_1^u, T_2^u, \ldots, T_n^u\}$.

When we are processing the i-th trajectory T_i^u, we need to take into account the previous historical trajectories of user u, which is called the trajectory context. We denote $C_i^u = \{T_1^u, T_2^u, \ldots, T_{i-1}^u\}$ as the context of the current trajectory T_i^u. When there is no ambiguity, we will omit the subscript u.

Formally, given a user u and all of their historical trajectories $T^u = \{T_1^u, T_2^u, \ldots, T_n^u\}$, the problem of making the next POI recommendation is to predict the next POI p_{n+1} that the user is most likely to visit.

3.2 PDPNN Model

Basic Framework. The PDPNN model receives current trajectory T_i and its trajectory context C_i as input. C_i is the previous trajectories of each trajectory T_i of the same user. The model learns short-term preference mainly from current trajectory T_i, and builds a personal dynamic preference from both T_i and its trajectory context C_i. The architecture of our proposed model PDPNN is shown in Fig. 1 :

- (I) Attention-enhanced recurrent neural network (ARNN) module: We use a LSTM network to capture short-term and long-term spatial-temporal sequence patterns. Additionally, an attention mechanism is introduced to capture the weight of all hidden states of the trajectory sequence. This component is used as an important part of the personal dynamic preference and the short-term preference learning module.
- (II) Personal dynamic preference learning module: The user's personal preferences is important for personalized POI recommendation, which is implicit in their historical trajectories. This module obtains a representation of personal dynamic preference from the user's current trajectory and trajectory context C_i. We first utilize the ARNN module to calculate the output hidden states of all trajectories in the trajectory context. Obviously, not every state can help to predict a POI, and the historical trajectories that are highly correlated with the current trajectory are supposed to have greater weight. To this end, we introduce attention mechanism into the module to measure the spatial temporal context similarity of the trajectory context and the current trajectory, and aggregate all the hidden states of the trajectory context as the personal dynamic preference representation.
- (III) Short-term preference learning module: The short-term preference, which is usually referred to as sequential preference, is commonly learned from the current trajectory. We simply apply the ARNN module described above to get the hidden state of the current trajectory T_i .
- (IV) Classifier: This is the final output component which unifies the user embedding output, the last hidden state from current trajectory and the aggregated attention from historical trajectories into a feature representation; after this it predicts the next POI. The output of this module is the probability vector of every POI that the user is likely to visit next. Cross-entropy loss and L2 regularization is used to measure the total loss in this module.

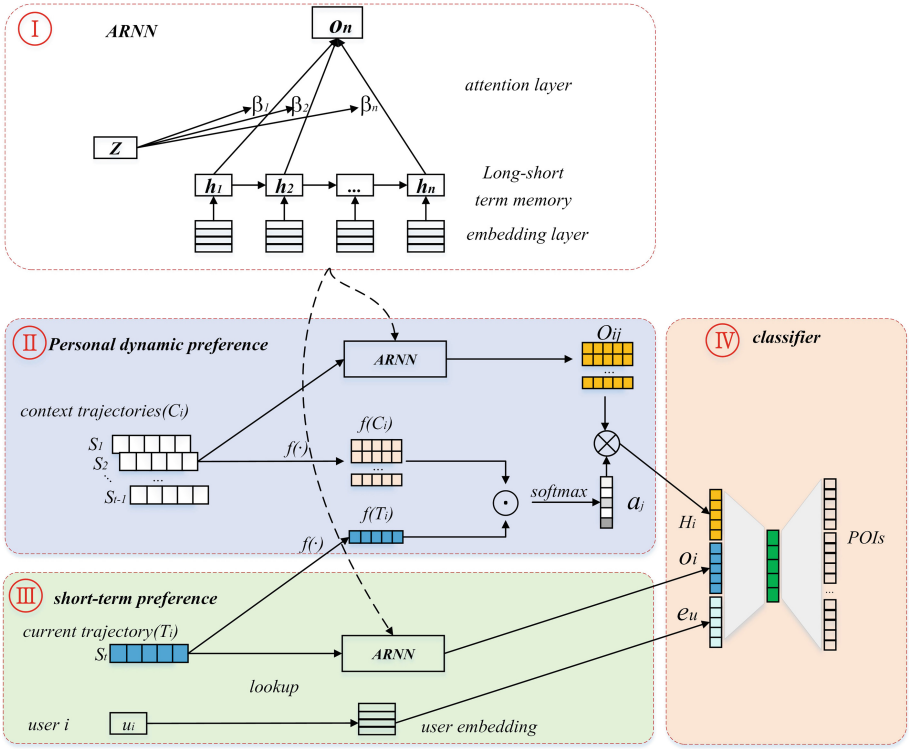

Fig. 1. The architecture of PDPNN model.

ARNN: ARNN is an attention-enhanced recurrent neural network, which receives a trajectory T^u as input and outputs the weighted sum of the hidden states of the user's trajectory via a attention mechanism. The ARNN model consists of an embedding layer, a recurrent layer and an attention layer.

Each input trajectory contains a sequence of POI identifier (id) p, time interval Δs, geographic distance Δt, longitude lo and latitude la. The input POI id is then transformed into a latent space vector e_i^p by the embedding layer. Input data I_t can be described as follows:

$$I_t = [e_t^p; \Delta s_t; \Delta t_t; la_t; lo_t] \tag{1}$$

where $\Delta s = \sqrt{(la_t - la_{t-1})^2 - (lo_t - lo_{t-1})^2}$, and $\Delta t = t_t - t_{t-1}$, the subscript $t \in [2, \|T^u\|]$.

We utilize a Long Short-Term Memory (LSTM) network in the recurrent layer. A LSTM neuron unit consists of an input gate i_t, an input gate f_t, and an output gate o_t. These parameters are explained in detail below:

$$f_t = \delta(W_f \cdot [h_{t-1}; I_t] + b_f) \tag{2}$$

$$i_t = \delta(W_i \cdot [h_{t-1}; I_t] + b_i) \tag{3}$$

$$\widetilde{c}_t = \delta(W_c \cdot [h_{t-1}; I_t] + b_c) \tag{4}$$

$$c_t = f_t \odot c_{t-1} + i_c \odot \widetilde{c}_t \tag{5}$$

$$o_t = \delta(W_o \cdot [h_{t-1}; I_t] + b_o) \tag{6}$$

$$h_t = o_t \odot tanh(c_t) \tag{7}$$

The attention mechanism aims to capture the importance of all the state sequences. Two kinds of attention mechanism have been proposed, known as additive attention [16] and dot-product attention [15]. Considering that an optimized matrix multiplication operation is much faster and more space-efficient in practice than an alignment calculation through hidden layer [15], we chose dot-product attention to calculate the attention weight of the state in the attention layer. Following the work of [15], we introduce Z_u as the user context, which can be learned during training. We get the weight of each hidden state β_t with a softmax function:

$$\beta_t = \frac{exp(h_t \cdot Z_u)}{\sum_{j=1}^{n} exp(h_j \cdot Z_u)} \tag{8}$$

We can get the weighted sum of all state sequences as the overall output.

The overall trajectory state output O_n is the weighted sum of all state sequences, which is described as follows:

$$O_n = \sum_{t=1}^{n} \beta_t \odot h_t \tag{9}$$

For the convenience of subsequent references, we summarize Eqs. (1)–(9) as function $ARNN(\cdot)$:

$$O_n = ARNN(T^u) \tag{10}$$

Personal Dynamic Preference Learning Module: Unlike routine RNN models that treat each separate trajectory as input, PDPNN attaches to each trajectory T_i the trajectory context $C_i = \{T_1, T_2, \ldots, T_{i-1}\}$ of the same user to produce the input.

Obviously, the length of the current trajectory processed by the padding operation is fixed, while the trajectory context can not be filled or truncated due to the large length changes. This represents a difficult problem in model training. To solve this problem, we separate the processing of trajectory context from the processing of the current trajectory. For each trajectory context C_i , we use the ARNN model to calculate the hidden state output of each trajectory T_j in C_i:

$$O_j^{(i)} = ARNN(T_j), j \in [1, i-1] \tag{11}$$

Trajectories that are highly relevant to the current trajectory are considered to play a more important role in guiding POI predictions for all historical trajectories in the trajectory context. To this end, we introduce a correlation function

$f(\cdot)$ to calculate the weight of each historical trajectory, and use softmax to calculate the weight of all historical trajectory states:

$$w_{ij} = \frac{exp(f(T_i^u, T_j^u))}{\sum_{j=1}^{i-1} exp(f(T_i^u, T_j^u))}, j \in [1, i-1] \tag{12}$$

Function $f(\cdot)$ here can be any function used to measure the correlation of trajectories. We use the Jaccard Similarity of the POI set of the trajectory to measure the similarity between the trajectories in the paper:

$$f(T_i^u, T_j^u) = \frac{|P_i \cap P_j|}{|P_i \cup P_j|}), j \in [1, i-1] \tag{13}$$

where P_i and P_j are the POI sets of T_i^u and T_j^u, respectively.

Then we multiply the weight of the context trajectories and their corresponding output latent states. The user's personal dynamic preference H_i is represented as the weighted sum of the products above:

$$H_i = \sum_{j=1}^{i-1} w_{ij} \cdot O_j^{(i)} \tag{14}$$

Short-Term Preference Learning Module: The short-term preference learning module is mainly used to capture the sequential POI transition preferences of users. The module consists of a user embedding layer and an ARNN model, and receives a user and trajectories as input. Each trajectory is represented by a sequence of POI identifier, time interval, geographic distance, longitude and latitude as desribed in Eq. (1). Users are represented by a user identifier u in real world data, which can not precisely reflect the similarities and differences between users. The appropriate representation of a user can accurately measure the similarities between users, so that the model can learn the similarity of user behavior preferences according to degree of similarity between users. To this end, we use a fully connected network to embed user identifiers into latent space.

$$e_u = tanh(W_u \cdot u + b_u) \tag{15}$$

where parameter $W_u \in \mathbf{R}^{\|U\| \times d_u}$ and $b_u \in \mathbf{R}^{1 \times d_u}$.

The output of the short-term preference learning module can be described as follows:

$$O_i = ARNN(T_i) \tag{16}$$

Classifier: We consider the next POI recommendation as a multiclass classification problem. The classifier module concatenates the user embedding, the user's personal dynamic preference and the current trajectory state as the input:

$$Q = [H_i; O_i; e_u] \tag{17}$$

Then, we feed Q into a fully connected layer with softmax function to calculate the probability of each POI:

$$\hat{y}_t = softmax(sigmoid(W_s \cdot Q + b_s)) \tag{18}$$

where parameter $W_s \in \mathbf{R}^{\|P\| \times d_p}$ and b_s is the bias parameter.

We adopt the cross-entropy loss between the ground truth POI y_t and the predicted POI $\hat{y}_t$. The loss function can be calculated as follows:

$$\begin{gathered} L(I_1^{(1)}, \cdots I_1^{(T^{(1)}-1)} \cdots, I_N^{(1)}, I_N^{(T^{(N)}-1)}, \theta) = \\ -\frac{1}{N} \sum_{n=1}^{N} y_t log(\hat{y}_t) + (1 - y_t) log(1 - \hat{y}_t) + \frac{\lambda}{2} \left\| \Theta^2 \right\| \end{gathered} \tag{19}$$

where Θ denotes the parameter set of PDPNN, and λ is a pre-defined hyper parameter for the L2 regularization to avoid overfitting. To optimize the above loss function, we use Stochastic Gradient Descent to learn the parameter set Θ.

3.3 Model Training

Training the model with short-term preference and personal dynamic preference jointly is time-consuming, so we use pre-trained model and inner epoch cache method to accelerate the training process.

Pre-trained Model: The ARNN is the core component of the PDPNN model, and it takes most of the time required by the training process. In other words, if the ARNN component could be trained as quickly as possible, the training speed of the whole model would be accelerated. To this end, we try to pre-train the ARNN component in a more simple model, which connects the output of the ARNN and the classifier directly. Then, we load the pre-trained model and modify its network structure to adapt to our current model in training.

Inner Epoch Cache: Each input of the PDPNN model contains the current trajectory and the trajectory context, which makes the training process time consuming. For a certain user u with n trajectories $\{T_1, T_2, \ldots, T_n\}$, the PDPNN needs to process all of the n trajectories and the corresponding trajectory contexts. For each trajectory T_i , the PDPNN learns a short-term preference from T_i, and learns the user's personal dynamic preference from its trajectory context $C_i = \{T_1, T_2, \ldots, T_{i-1}\}$, which means the PDPNN needs to calculate the output state of C_i with ARNN component for i times. So, the whole training complexity of the n trajectories user u is $O(n^2)$.

As a part of trajectory contexts of $T_{i+1}, T_{i+2}, \ldots, T_n$, the i-th trajectory T_i is repeatedly calculated for $n - i$ times during each training epoch. The optimization we can intuitively think of is to eliminate the repeated calculations. However, trajectory context state values are updated during training synchronously,

which means we can not simply cache the state value in the whole training process. Considering the increment of state value updates between batches is very small in each training epoch, if the trajectory context output states can be cached within one training epoch and updated between epochs, a large number of approximately repeated calculations can be eliminated. To this end, we apply the inner epoch cache to store the trajectory context output state value during model training, and the calculation of each user's personal dynamic preference can be reduced to linear complexity.

4 Experiment Analysis and Evaluation

4.1 Experiment Settings

Datasets: We conducted experiments on three publicly available LBSN datasets, NYC, TKY and CA. NYC and TKY [18] are two datasets collected from users sharing their check-ins on the Foursquare website in New York and Tokyo, respectively. CA is a subset of a Foursquare dataset [19], which includes long-term, global-scale check-in data. We chose the check-ins of users in California for the dataset in this paper. The check-in times of above datasets range from 2012 Apr to 2013 Sep. Each record contains an anonymous user identifier, time, POI-id, POI category, latitude and longitude of the check-in behavior. In order to alleviate the problem of data sparsity, following previous work [12], we expand the set of trajectories by adding the sub-trajectories of the original trajectories. For all datasets, we choose 90% of each user's trajectories as the training set, and the remaining 10% as testing data.

Table 1. The statistics of datasets.

Dataset	#Users	#check-ins	#location	#trajectories
LA	1,083	22,7428	38,333	31,941
TKY	2,293	57,3703	61,858	66,663
CA	4,163	48,3805	2,9529	47,276

Comparing Methods: We compare PDPNN with several representative methods for location prediction:

- RNN [7]: This is a basic method for POI prediction, which has been successfully applied in word embedding and ad click prediction.
- AT-RNN: This method empowers the RNN model with an attention mechanism, which has been successfully applied in machine translation and vision processing.
- LSTM [14]: This is a variant of the RNN model, which contains a memory cell and three multiplicative gates to allow long-term dependency learning.

– ATST-LSTM [9]: This is a state-of-the-art method for POI prediction, which applies an attention mechanism to a LSTM network.
– PDPNN: This is our approach, which learns a user's personal dynamic preference base on the current spatial and temporal context.

Parameter Settings: The key hyper parameters in PDPNN include: (1) the embedding dimension for POI and user, namely d_u and d_p; (2) the dimension d_h for the hidden state; (3) the regularization parameter λ. In general, the performance of the PDPNN increases with the above dimensions and gradually stabilizes when dimensions are large enough. In our experiments, we finally set $d_u = d_p = d_h = 200$. For the regularization parameter λ, we tried values in $\{1,0.1,0.01,0.001\}$ and $\lambda = 0.01$ turns out to have the best performance.

Table 2. Prediction performance comparison on three dataset.

Dataset	Method	Recall@1	Recall@5	Recall@10	Recall@20
LA	RNN	0.7901%	3.0536%	5.2317%	9.4597%
	AT-RNN	1.9218%	7.4098%	14.4352%	24.685%
	LSTM	6.8332%	25.71%	39.0562%	50.5659%
	ATST-LSTM	7.9650%	26.009%	39.9317%	51.5909%
	PDPNN	**8.5629%**	**28.2084%**	**41.0634%**	**52.0606%**
TKY	RNN	2.5639%	7.2056%	9.4411%	11.8282%
	AT-RNN	4.0859%	16.8172%	26.8203%	36.9498%
	LSTM	2.6271%	7.2182%	9.8958%	13.5965%
	ATST-LSTM	4.4143%	18.5475	**28.9233%**	**38.8317%**
	PDPNN	**4.8058%**	**18.5665%**	28.2918%	37.0445%
CA	RNN	0.5711%	2.4112%	4.3993%	8.0795%
	AT-RNN	6.5144%	17.8299%	24.6616%	32.0643%
	LSTM	2.0093%	7.0008%	10.7445%	15.4399%
	ATST-LSTM	6.6836%	17.7453%	24.4078%	32.1489%
	PDPNN	**7.1912%**	**18.6548%**	**25.5499%**	**32.9315%**

Metrics: To evaluate the performance of all methods for the POI recommendation problem, we employ a commonly used metric known as recall@N. The recall@N metric is popular in ranking tasks, which evaluates where the ground truth next POI appears in a ranked prediction list. A larger metric value indicates better performance.

4.2 Comparison of Recommendation Performance

We conducted experiments on a machine with Intel a Xeon CPU and a NVIDIA Tesla P4 GPU. The performance comparison of methods on three datasets is

illustrated in Table 2. The RNN has a lower baseline performance than other methods with no extra optimization. The AT-RNN improves the performance greatly by 2 or 3 times over the RNN on all three datasets, which shows that the attention mechanism can alleviate the problem of long-term dependencies with gradient descent of RNN. The LSTM outperforms the RNN largely because of its memory and forget gate design, but fails to outperform the AT-RNN for the TKY and CA datasets. ATST-LSTM turns out to be the strongest baseline and shows significant improvement on the TKY and CA datasets. The reason is that ATST-LSTM works better at learning long-term dependencies through the gate and attention mechanisms. This shows that the attention mechanism is an effective supplement to LSTM models.

PDPNN outperforms the above baseline methods in almost all of the recall metrics. Compared with ATST-LSTM, the PDPNN obtains recall@1, recall@5 and recall@10 improvements of 7–10% on LA and CA, and recall@1 and recall@5 improvements of 5–7% on TKY. By creating an elaborate modeling of users' personal dynamic preference, the PDPNN can capture the intentions of user activities more accurately. This enables the PDPNN to achieve better performance in next POI recommendation. It is worth noting that the PDPNN has lower performance in recall@10 and recall@20 than the ATST-LSTM on the TKY dataset. The reason is that the PDPNN treats POI recommendation as a classification problem, and the objective function is to optimize the accuracy of recall@1. Besides, there may be another interesting reason that people's preferences vary widely from country to country.

4.3 Inner Epoch Cache Evaluation

To evaluate the efficiency of the inner epoch cache, we compare the average time consumption of models with and without the inner epoch cache during one epoch training. Figure 2 shows that the PDPNN with an inner epoch cache

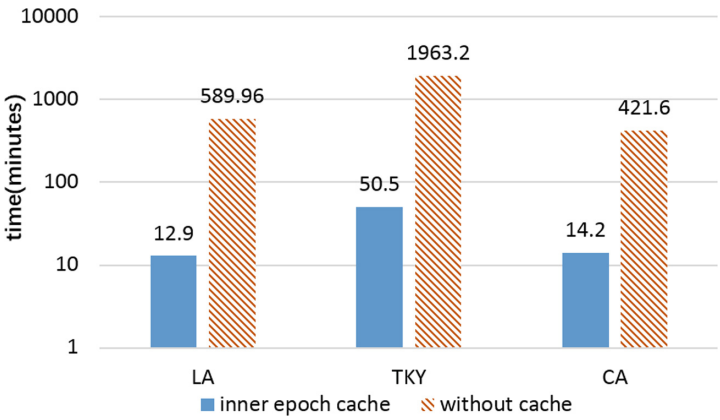

Fig. 2. Time-consuming comparison in one training epoch.

trains 45×, 38× and 30× faster than the model without a cache on LA, TKY and CA dataset, respectively. By introducing the inner epoch cache optimization in training, the per-epoch training time is reduced from more than one day to one hour on the TKY data set. It shows that the inner epoch cache can greatly accelerate the model training process.

5 Conclusion

In this paper, we have proposed a new approach to better model the user's personal dynamic preference of next POI from current trajectory supplemented with a trajectory context. Based on an attention empowered LSTM neural network, we build a new neural network model named PDPNN for next POI recommendation. Moreover, to accelerate the training of trajectory contexts, we proposed an inner epoch cache to store the trajectory context output state value during model training, and reduce the complexity of the user's personal dynamic preference module from $O(n^2)$ to $O(n)$ in each training epoch. We conduct experiments on three real world data set and show that our model outperforms current well-known methods.

Acknowledgment. This work was supported by Beijing Municipal Science & Technology Commission (Z191100007119003).

References

1. Gambs, S., Killijian, M.-O., del Prado Cortez, M.N.: Next place prediction using mobility markov chains. In: MPM (2012)
2. Steffen, R., Christoph, F., Lars, S.T.: Factorizing personalized Markov chains for nextbasket recommendation. In: WWW 811–820 (2010)
3. Yehuda Koren; Robert Bell; and Chris Volinsky: Matrix factorization techniques for recommender systems. IEEE Comput. **42**(8), 30–37 (2009)
4. Xiong, L., Chen, X., Huang, T.-K., Schneider, J., Carbonell, J.G.: Temporal collaborative filtering with bayesian probabilistic tensor factorization. In: SDM, pp. 211–222 (2010)
5. Lian, D., Zhao, C., Xie, X., Sun, G., Chen, E., Yong, R.: GeoMF, joint geographical modeling and matrix factorization for point-of-interest recommendation. In: KDD (2014)
6. Zheng, V.W., Cao, B., Zheng, Y., Xie, X., Yang, Q.: Collaborative filtering meets mobile recommendation, a user-centered approach. In: AAAI (2010)
7. Zhang, Y., et al.: Sequential click prediction for sponsored search with recurrent neural networks. In: AAAI, pp. 1369–1376 (2014)
8. Liu. Q., Wu, S., Wang, L., Tan, T.: Predicting the next location : a recurrent model with spatial and temporal contexts. In: AAAI 2016 (2016)
9. Liwei, H., Yutao, M., Shibo, W., Yanbo, L.: An attention-based spatio-temporal LSTM network for next POI recommendation. In: IEEE Transactions on Services Computing, p. 1 (2019). https://doi.org/10.1109/TSC.2019.2918310
10. Zhao, P., Zhu, H., Liu, Y., Li, Z., Xu, J., Victor, S.: Where to go next: a spatio-temporal LSTM model for next POI recommendation. In: AAAI 2019 (2019)

11. Yao, D., Zhang, C., Huang, J., Bi, J.: SERM: a recurrent model for next location prediction in semantic trajectories. In: CIKM (2017)
12. Jannach, D., Lerche, L., Jugovac, M.: Adaptation and evaluation of recommendations for short-term shopping goals. In: RecSys, Adaptation and evaluation of recommendations for short-term shopping goals. pp. 211–218 (2015)
13. Bengio, Y., Frasconi, P., Simard, P.: Learning long-term dependencies with gradient descent is difficult. IEEE Trans. Neural Netw. **5**(2), 157–166 (1994)
14. Schuster, M., Paliwal, K.P.: Bidirectional recurrent neural networks. IEEE Trans. Signal Process. **45**(11), 2673–2681 (1997)
15. Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., Hovy, E.: Hierarchical attention networks for document classification. In: Proceedings of NAACL (2016)
16. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: Proceedings of ICLR (2015)
17. Vaswani, A., et al.: Attention is all you need. In: Proceedings of NIPS (2017)
18. 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. (TSMC) **45**(1), 129–142 (2015)
19. Yang, D., Zhang, D., Qu, B.: Participatory cultural mapping based on collective behavior data in location based social networks. ACM Trans. Intell. Syst. Technol. (TIST) **7**(3), 30 (2016)