Location Prediction over Sparse User Mobility Traces Using RNNs: Flashback in Hidden States!

Dingqi Yang[1,3], **Benjamin Fankhauser**[2], **Paolo Rosso**[1] and **Philippe Cudre-Mauroux**[1]
[1]University of Fribourg, Switzerland
[2]University of Bern, Switzerland
[3]University of Macau, SAR China
dingqi.yang@unifr.ch, benjamin.fankhauser@students.unibe.ch, {paolo.rosso, pcm}@unifr.ch

Abstract

Location prediction is a key problem in human mobility modeling, which predicts a user's next location based on historical user mobility traces. As a sequential prediction problem by nature, it has been recently studied using Recurrent Neural Networks (RNNs). Due to the sparsity of user mobility traces, existing techniques strive to improve RNNs by considering spatiotemporal contexts. The most adopted scheme is to incorporate spatiotemporal factors into the recurrent hidden state passing process of RNNs using context-parameterized transition matrices or gates. However, such a scheme oversimplifies the temporal periodicity and spatial regularity of user mobility, and thus cannot fully benefit from rich historical spatiotemporal contexts encoded in user mobility traces. Against this background, we propose Flashback, a general RNN architecture designed for modeling sparse user mobility traces by doing *flashbacks* on hidden states in RNNs. Specifically, Flashback explicitly uses spatiotemporal contexts to search past hidden states with high predictive power (i.e., historical hidden states sharing similar contexts as the current one) for location prediction, which can then directly benefit from rich spatiotemporal contexts. Our extensive evaluation compares Flashback against a sizable collection of state-of-the-art techniques on two real-world LBSN datasets. Results show that Flashback consistently and significantly outperforms state-of-the-art RNNs involving spatiotemporal factors by 15.9% to 27.6% in the next location prediction task.

1 Introduction

User mobility modeling is one of the most important problems in understanding human dynamics, which also serves as a fundamental ingredient for developing smart city applications. One key task of user mobility modeling is to predict a user's next location based on users' historical mobility traces [Noulas *et al.*, 2012]. Traditional methods often resort to user mobility features — either hand-crafted features such as historical visit counts [Noulas *et al.*, 2012], or automatically-learnt features using graph embedding techniques [Xie *et al.*, 2016]) — to capture user mobility patterns. However, by generating static features from historical data, these techniques predict user locations without really considering the sequential patterns of user mobility, which have been shown as an important clue for location prediction [Liu *et al.*, 2016].

Recently, Recurrent Neural Networks (RNNs) have been shown as a successful tool to model sequential data and thus started to be used also for user mobility modeling [Zhao *et al.*, 2019]. However, classical RNN architectures, such as vanilla RNN, Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), were originally designed for language modeling to learn from word sequences (sentences) and they are not able to handle sparse (and incomplete) mobility traces. For example, on Foursquare, one of the most popular Location Based Social Networks (LBSNs), a user's mobility trace is stored as a sequence of check-ins, where each check-in represents the user's presence at a specific Point of Interests (POIs) such as a restaurant or a gym, at a specific time. As users share their check-ins on a *voluntary* basis on this platform, such mobility traces are often sparse; on our collected Foursquare dataset, we find that the average time between successive check-ins are about 59 hours. Such sparsity and incompleteness of input sequences hinders the application of RNNs to the location prediction problem [Feng *et al.*, 2018]. To handle such sparse mobility traces, existing works strive to incorporate spatiotemporal factors into RNN architectures, as spatiotemporal contexts have indeed been shown as strong predictors for user mobility prediction [Yang *et al.*, 2015].

In the current literature, the most popular scheme to achieve this goal is to incorporate spatiotemporal factors into the recurrent hidden state passing process of RNNs. Specifically, given a user mobility trace represented as a sequence of POIs $\{..., p_{i-1}, p_i, p_{i+1}, ...\}$, a classical RNN learns from the sequence by outputting a hidden state h_i from the current POI p_i and the previous hidden state h_{i-1}, i.e., $h_i = \mathcal{F}(p_i, h_{i-1})$ where $\mathcal{F}(\cdot)$ denotes a RNN unit (e.g., vanilla RNN, LSTM or GRU). Subsequently, to add spatiotemporal factors, existing techniques first compute the temporal and spatial distances between the previous check-in and the current check-in, denoted as $\Delta T_{i-1,i}$, $\Delta D_{i-1,i}$, respectively, and then feed them as additional inputs to the RNN unit, i.e., $h_i = \mathcal{F}(p_i, h_{i-1}, \Delta T_{i-1,i}, \Delta D_{i-1,i})$, as illustrated in Figure 1. In the current literature, this scheme

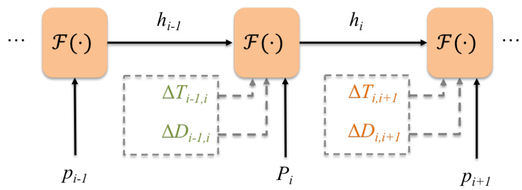
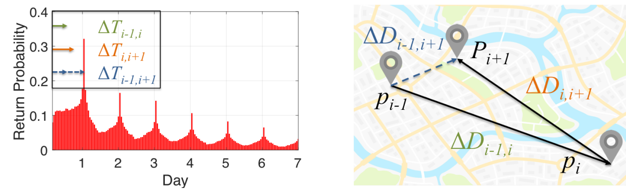


Figure 1: The most popular scheme adding spatiotemporal factors into RNNs. While a classical RNN unit outputs $h_i = \mathcal{F}(p_i, h_{i-1})$, a spatiotemporal RNN outputs $h_i = \mathcal{F}(p_i, h_{i-1}, \Delta T_{i-1,i}, \Delta D_{i-1,i})$, where $\Delta T_{i-1,i}$ and $\Delta D_{i-1,i}$ denote the temporal and spatial distances between the previous check-in and the current check-in.



(a) (b)

Figure 2: Spatiotemporal factors in user check-in data. a) Temporal factor shown as periodicity, where the example in the box shows that considering temporal distances between successive check-ins only ($\Delta T_{i-1,i}$ and $\Delta T_{i,i+1}$) cannot capture such a periodicity. b) Spatial factor where the example shows that considering spatial distances between *successive* check-ins only ($\Delta D_{i-1,i}$ and $\Delta D_{i,i+1}$) cannot capture the proper distances $\Delta D_{i-1,i+1}$. In both cases, the corresponding techniques fail to fully benefit from the historical check-ins with high predictive power when predicting location.

has been instantiated by either using spatiotemporal-specific transition matrices parameterized by $\Delta T_{i-1,i}$ and $\Delta D_{i-1,i}$ in RNNs [Liu *et al.*, 2016], or extending/adding gates controlled by $\Delta T_{i-1,i}$ and $\Delta D_{i-1,i}$ to LSTM [Kong and Wu, 2018; Zhao *et al.*, 2019]. However, this scheme cannot fully benefit from rich historical spatiotemporal contexts encoded in the mobility trace, due to the following reasons.

First, from a temporal perspective, feeding temporal distances between successive check-ins to the RNN unit may ignore the *temporal periodicity* of user mobility. Specifically, the periodicity of human activities is universal [Gonzalez *et al.*, 2008]. Figure 2(a) shows the return probability of user check-ins over time, defined as the probability of a user re-checking in at a POI a certain period of time after her first check-in at that POI, on our collected Foursquare dataset. We observe a clear daily (periodic) revisiting pattern. In the context of location prediction, such a periodicity implies that historical check-ins with a temporal distance (taken from the current time) closer to these daily peaks (1 day, 2 days, etc.) have higher predictive power. However, iteratively feeding temporal distances between *successive* check-ins into the RNN unit (as shown in Figure 1) often fails to benefit from this periodicity property. Figure 2(a) illustrates such an example, where $\Delta T_{i-1,i}$ and $\Delta T_{i,i+1}$ refers to the temporal distances between successive check-in pairs $p_{i-1} \to p_i$ and $p_i \to p_{i+1}$, respectively. If we consider the temporal distance in two steps ($p_{i-1} \to p_{i+1}$), we find that $\Delta T_{i-1,i+1}$ is close to the 1 day peak, indicating that p_{i-1} is very helpful for predicting the next location. In contrast, this cannot be captured if we iteratively feed temporal distances $\Delta T_{i-1,i}$ and $\Delta T_{i,i+1}$ into the RNN unit.

Second, from a spatial perspective, feeding spatial distances between successive check-ins to the RNN unit oversimplifies the *spatial regularity* of user mobility. Specifically, it has been found that a user's check-ins in a region that she frequently visited are highly biased to certain POIs [Yang *et al.*, 2015]. In other words, those regions often have certain implicit "functions", such as working or shopping. Subsequently, the closer the user is to such a region, the more predictable her behavior is. This suggests that the closer a past check-in is located to the current location, the more helpful it is for the next location prediction. However, only considering spatial distances between successive check-ins fails to capture such distances over space. Figure 2(b) shows an example. We observe that the spatial distance in two steps $\Delta D_{i-1,i+1}$ is much smaller than both $\Delta D_{i-1,i}$ and $\Delta D_{i,i+1}$, suggesting that p_{i-1} is more helpful for location prediction. In contrast, this cannot be captured if we consider spatial distances $\Delta D_{i-1,i}$ and $\Delta D_{i,i+1}$ only.

Against this background, we propose Flashback, a general RNN architecture designed for modeling sparse user mobility traces, with a particular consideration on leveraging rich historical spatiotemporal contexts by considering flashbacks on hidden states in RNNs. More precisely, departing from the widely adopted scheme of adding spatiotemporal factors into the recurrent hidden state passing process of the RNNs, our solution *explicitly* uses the spatiotemporal context to search past hidden states with high predictive power (i.e., historical hidden states that share similar spatiotemporal contexts to the current one) for predicting the next location; as a result, our scheme can directly benefit from rich temporal (i.e, periodicity as shown in Figure 2(a)) and spatial (i.e., distances over space as shown in Figure 2(b)) contexts encoded in user mobility traces. Moreover, as we do not modify the hidden state passing process of RNNs (while many existing techniques do), our Flashback can be easily instantiated with any RNN units (e.g., vanilla RNN, LSTM or GRU). We conduct a thorough evaluation of our method compared to a sizable collection of baselines on two real-world LBSN datasets. Results show that Flashback consistently and significantly outperforms all baseline techniques. In particular, it yields an improvement of 15.9% to 27.6% over the best performing spatiotemporal RNNs.

2 Related Work

Location prediction is a key problem in human mobility modeling, which predicts the location of a user based on the user's historical mobility traces. Traditional methods for location prediction often resort to various mobility features, such as hand-craft features including historical visit counts [Noulas *et al.*, 2012; Yang *et al.*, 2016] and activity preferences [Yang *et al.*, 2015], or automatically-learnt features using graph embedding techniques [Xie *et al.*, 2016; Yang *et al.*, 2019]. In addition, generative/factorization models have also been used to solve location prediction/recommendation problems [Kurashima *et al.*, 2013; Yang *et al.*, 2013a; 2013b]. However, these techniques have intrinsic limitations when capturing the sequential patterns of user mobility.

To capture user sequential mobility patterns, (Hidden) Markov Chains have been widely used for sequential prediction [Mathew *et al.*, 2012; Cheng *et al.*, 2013; Feng *et al.*, 2015]. The basic idea is to estimate a transition matrix encoding the probability of a behavior based on previous behaviors. A typical technique here is Factorizing Personalized Markov Chains (FPMC) [Rendle *et al.*, 2010], which estimates a personalized transition matrix via matrix factorization techniques. FPMC has been extended to the location prediction problem by further considering spatial constraints [Cheng *et al.*, 2013; Feng *et al.*, 2015] in building the transition matrices.

Recently, Recurrent Neural Networks (RNNs) have been shown as a successful tool to model sequential data [Mikolov *et al.*, 2010], capturing complex long- and short-term dependency over input sequences. To handle sparse and incomplete sequences, existing techniques strive to add context factors into the RNNs. For example, temporal factors can be added by truncating each sparse input sequence into several short sessions [Hidasi *et al.*, 2016; Feng *et al.*, 2018], or by considering temporal factors as additional inputs to the RNN units [Neil *et al.*, 2016; Zhu *et al.*, 2017]. For the problem of location prediction over sparse user mobility sequences, spatiotemporal factors have been shown as strong predictors [Yang *et al.*, 2015]. The most popular scheme to incorporate spatiotemporal factors into RNNs is adding the spatiotemporal distances between (mostly successive) check-ins as additional inputs to the RNN units (as illustrated in Figure 1). For example, STRNN [Liu *et al.*, 2016] uses spatiotemporal-specific transition matrices parameterized by the spatiotemporal distances in RNNs; HST-LSTM [Kong and Wu, 2018] extends existing gates in LSTMs to let these gates take the spatiotemporal distance as an additional input; STGN [Zhao *et al.*, 2019] adds additional gates controlled by the spatiotemporal distances to LSTMs. However, as discussed in the Introduction, such schemes cannot fully benefit from the rich historical spatiotemporal contexts encoded in mobility traces. Therefore, we propose in this paper Flashback, a general RNN architecture that explicitly uses spatiotemporal contexts to search past hidden states with high predictive power for location prediction, in order to directly benefit from the rich spatiotemporal contexts encoded in user mobility traces.

3 Flashback

Flashback is designed for modeling sparse user mobility traces, with a particular focus on leveraging rich spatiotemporal contexts by doing flashbacks on hidden states in RNNs. Instead of implicitly considering context factors by adding spatiotemporal factors into the recurrent hidden state passing process of RNNs (as most existing techniques do), our solution explicitly uses the spatiotemporal contexts to search past hidden states with high predictive power (i.e., historical hidden states that share similar contexts as the current one) for location prediction; it can thus directly benefit from rich spatiotemporal contexts encoded in user mobility traces.

3.1 Overview

Figure 3 shows an overview of our Flashback architecture. Given a user's mobility trace represented as a POI sequence $\{..., p_{i-3}, p_{i-2}, p_{i-1}, p_i, p_{i+1}...\}$, we denote the temporal and spatial distances between two check-ins p_i and p_j as $\Delta T_{i,j}$ and $\Delta D_{i,j}$, respectively. As shown in Figure 3, our recurrent hidden state passing process remains unaltered from classical RNNs i.e., $h_i = \mathcal{F}(p_i, h_{i-1})$, letting RNNs capture sequential user mobility patterns. However, instead of using only the current hidden state h_i to predict the next location p_{i+1} (as classical RNNs do), we leverage the spatiotemporal context to search past hidden states with high predictive power. To achieve this goal, we compute the weighted average of the historical hidden states h_j, $j < i$, with a weight $\mathcal{W}(\Delta T_{i,j}, \Delta D_{i,j})$ as an aggregated hidden state. This weight is parameterized by the temporal and spatial distances ($\Delta T_{i,j}$ and $\Delta D_{i,j}$, respectively) between check-in p_i and p_j and measures the predictive power of the hidden state h_j (more information on this point below). Finally, in order to model individual users preferences, we define a learnable user embedding vector for each user, which is concatenated with the aggregated hidden state and then fed into a fully connected layer for predicting the next location, as shown in Figure 3.

In summary, to effectively predict locations from sparse user mobility traces, Flashback 1) uses RNNs to capture *sequential* patterns, 2) leverages *spatiotemporal* contexts to search past hidden states with high predictive power, and 3) incorporates user embeddings to consider users *preferences*.

3.2 Context-Aware Hidden State Weighting

The weight $\mathcal{W}(\Delta T_{i,j}, \Delta D_{i,j})$ is designed to measure the predictive power of the hidden state h_j according to its spatiotemporal contexts.

First, from a temporal perspective, our primary goal is to incorporate the periodicity property of user behavior (as shown in Figure 2(a)) into $\mathcal{W}(\Delta T_{i,j}, \Delta D_{i,j})$. Specifically, we use a Havercosine function, a typical periodic function with outputs bounded in $[0, 1]$, parameterized by $\Delta T_{i,j}$ (in days) as follows:

$$w_{period}(\Delta T_{i,j}) = \text{hvc}(2\pi \Delta T_{i,j}) \tag{1}$$

where $\text{hvc}(x) = \frac{1+\cos(x)}{2}$ is the Havercosine function modeling the daily periodicity. Moreover, as we can see from Figure 2(a), the return probability exponentially decreases when increasing $\Delta T_{i,j}$, which indicates that besides the periodicity, the older a check-in is, the less impact it has for prediction. Subsequently, we add a temporal exponential decay weight to model this factor:

$$\begin{aligned} w_T(\Delta T_{i,j}) &= w_{period}(\Delta T_{i,j}) \cdot e^{-\alpha \Delta T_{i,j}} \\ &= \text{hvc}(2\pi \Delta T_{i,j}) \cdot e^{-\alpha \Delta T_{i,j}} \end{aligned} \tag{2}$$

where α is a temporal decay rate, controlling how fast the weight decreases over time $\Delta T_{i,j}$.

Second, from a spatial perspective, we consider the fact that the closer a check-in is to the current location, the more helpful it is for location prediction (as shown in Figure 2(b)). Accordingly, we use a distance exponential decay weight to model this factor:

$$w_S(\Delta D_{i,j}) = e^{-\beta \Delta D_{i,j}} \tag{3}$$

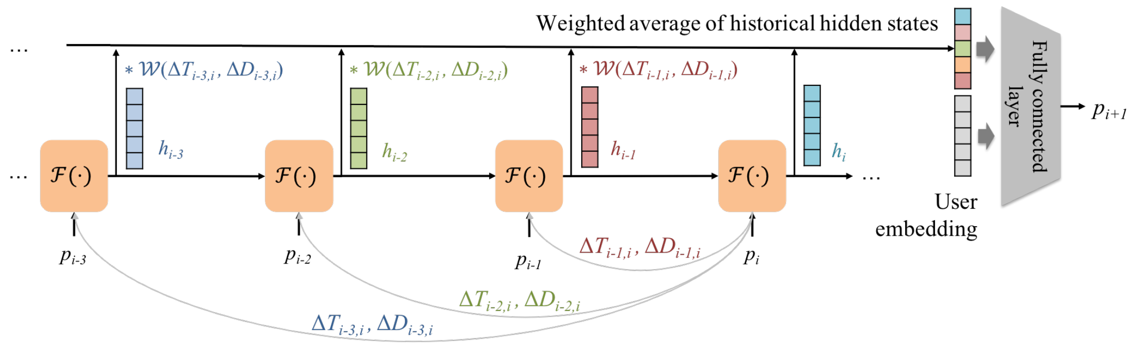


Figure 3: Overview of our Flashback architecture for next location prediction

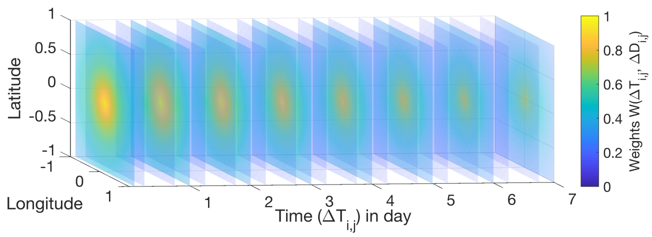
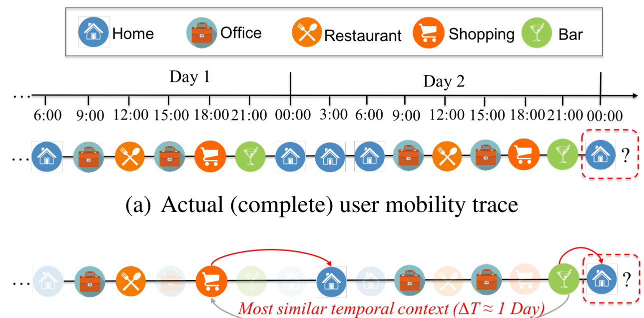


Figure 4: Visualization of our proposed weight $\mathcal{W}(\Delta T_{i,j}, \Delta D_{i,j})$ over space and time. $\Delta D_{i,j}$ is illustrated as L2 distance from the origin over spatial space (latitude and longitude axes). We observe both a periodicity pattern over time ($\Delta T_{i,j}$) and a spatiotemporal decay over space and time. The transparency of the slices on the time axis is proportional to the weights (the lower the weight is, the more transparent a slice is).



(a) Actual (complete) user mobility trace

(b) Observed (sparse) user mobility trace

Figure 5: A toy example illustrating the working principle of Flashback from a temporal perspective.

where $\Delta D_{i,j}$ is the L2 distance between the GPS coordinates of POIs p_i and p_j, and β is a spatial decay rate, controlling how fast the weight decreases over spatial distance $\Delta D_{i,j}$.

Finally, we obtain the weight $\mathcal{W}(\Delta T_{i,j}, \Delta D_{i,j})$ by combining the temporal and spatial weights together:

$$\begin{aligned}\mathcal{W}(\Delta T_{i,j}, \Delta D_{i,j}) &= w_T(\Delta T_{i,j}) \cdot w_S(\Delta D_{i,j}) \\ &= \mathrm{hvc}(2\pi \Delta T_{i,j}) e^{-\alpha \Delta T_{i,j}} e^{-\beta \Delta D_{i,j}}\end{aligned} \tag{4}$$

where the first Havercosine term captures the periodicity property of user check-ins, and the exponential terms model the spatiotemporal decay of the impact of historical check-ins on location prediction. Figure 4 offers a visualization of the weights over space and time.

3.3 Discussions

Why Does Flashback Work?

By flashing back to the historical hidden states, our Flashback can discount the *"noise"* from the recurrent hidden state passing process of the RNNs over *sparse* user mobility traces, and also create an explicit "attention" mechanism by leveraging past hidden states with high predictive power (i.e., historical hidden states that share similar contexts as the current hidden state) for location prediction. Figure 5 shows a toy example from the temporal perspective. On one hand, Figure 5(a) shows an actual (complete) user mobility trace with a clear sequential pattern ("Home-Office-Restaurant-Office-Shopping-Bar-Home"), where classical RNNs can effectively capture such a pattern and predict the next location "Home". On the other hand, for an observed (sparse) user mobility trace as shown in Figure 5(b), the sequential pattern is difficult to be captured by RNNs, where the hidden state passing process becomes noisy due to the incompleteness of the sequence. Even when considering temporal distances between successive check-ins as additional inputs of the RNN units (as many state-of-the-art techniques do), it still falls short in capturing long-term temporal (i.e, periodicity) dependencies. However, by flashing back to the historical hidden states sharing a similar temporal context as the current one ($\Delta T \approx 1$ day capturing the daily periodicity, i.e., at the similar time on the previous day), Flashback can predict the next location "Home".

How Far to Flash Back?

As Flashback generates an aggregated hidden state from past hidden states, an immediate question here is *how many past hidden states should be considered?* To answer this question, we review the temporal exponential decay term $e^{-\alpha \Delta T_{i,j}}$ in $\mathcal{W}(\Delta T_{i,j}, \Delta D_{i,j})$. Let $\overline{\Delta t}$ denote the average time between successive check-ins. The hidden state k-step back has an average $\overline{\Delta T} = \overline{\Delta t} \cdot k$, corresponding to a temporal exponential decay term $e^{-\alpha \overline{\Delta t} \cdot k}$. In practice, our empirical analysis shows that the average time between successive check-ins Δt are 51.28 (2.13 days) hours and 58.59 hours (2.44 days) on our Gowalla and Foursquare datasets, respectively (see Section 4.1), and the optimal temporal decay rate is 0.1 in our experiments on both datasets (see Section 4.3). Therefore, the temporal exponential decay term becomes a function of k. When setting $k = 20$ for example, this term is around

Dataset	Gowalla	Foursquare
#Users	52,979	46,065
#POIs	121,851	69,005
#Checkins	3,300,986	9,450,342
Collection period	02/2009-10/2010	04/2012-01/2014
Average time between successive check-ins	51.28 hours (2.13 days)	58.59 hours (2.44 days)

Table 1: Statistics of the datasets

0.01, which is indeed the upper bound of $\mathcal{W}(\Delta T_{i,j}, \Delta D_{i,j})$ as all other terms in $\mathcal{W}(\Delta T_{i,j}, \Delta D_{i,j})$ are bound to $[0, 1]$. In other words, a hidden state more than k-step back receives on average a weight less than 0.01, which contributes little to the aggregated hidden state. We also study this point in our experiments, where prediction performance flattens out when $k \geq 20$ (see Section 4.4).

4 Experiments

4.1 Experimental Setup

Dataset

We conduct experiments on two widely used check-in datasets collected from two LBSNs: Gowalla and Foursquare, respectively. Table 1 shows the statistics of the datasets. We chronologically split all the mobility traces into 80% for training and 20% for test.

Baselines

We compare Flashback against a sizable collection of state-of-the-art techniques from five categories:

- *User Preference-based Methods*: 1) **WRMF** [Hu *et al.*, 2008] learns user preferences on POIs using matrix factorization; 2) **BPR** [Rendle *et al.*, 2009] learns user preferences on POIs by minimizing a pairwise ranking loss.
- *Feature-based methods*: 1) Most Frequent Time (**MFT**, the best-performing feature by [Gao *et al.*, 2012]) ranks a POI according to a user's historical check-in count at a POI and at a specific time slot (24 hours in a day) in the training dataset; 2) **LBSN2Vec** [Yang *et al.*, 2019] learns user, time and POI feature vectors from a LBSN hypergraph, and ranks a POI according to its similarities with user and time in the feature space.
- *Markov-Chain-based Methods*: 1) **FPMC** [Rendle *et al.*, 2010] estimates a personalized transition matrix via matrix factorization techniques; 2) **PRME** [Feng *et al.*, 2015] learns user and POI embeddings to capture the personalized POI transition patterns. 3) **TribeFlow** [Figueiredo *et al.*, 2016] uses a semi-Markov chain model to capture the transition matrix over a latent environment.
- *Basic RNNs*: 1) **RNN** [Zhang *et al.*, 2014] is a vanilla RNN achitecture; 2) **LSTM** [Hochreiter and Schmidhuber, 1997] is capable of learning long-term dependency using a memory cell and multiplicative gates; 3) **GRU** [Cho *et al.*, 2014] captures long-term dependency by controlling information flow with two gates.
- *Spatiotemporal RNNs*: 1) **DeepMove** [Feng *et al.*, 2018] adds an attention mechanism to GRU for location prediction over sparse mobility traces; 2) **STRNN** [Liu *et al.*, 2016] uses customized transition matrices parameterized by the spatiotemporal distances between check-ins within a time window in RNNs; 3) **STGN** [Zhao *et al.*, 2019] add additional gates controlled by the spatiotemporal distances between successive check-ins to LSTM, while **STGCN** is a variant of STGN with coupled input and forget gates for improved efficiency.

For our proposed Flashback, as it does not depend on any specific RNN units, we instantiate it using all the three basic RNNs, named as **Flashback (RNN)**, **Flashback (LSTM)**, and **Flashback (GRU)**. We train Flashback by backpropagation through time using the Adam stochastic optimizer with cross entropy loss. We implement Flashback in PyTorch, and our code and datasets are available here[1].

Evaluation Protocol and Metrics

We evaluate Flashback in the next location prediction task, where we predict where a user will go next, given a sequence of her historical check-ins, as shown in Figure 3. We report two widely used metrics for location prediction: average Accuracy@N (Acc@N), where $N = 1, 5, 10$, and Mean Reciprocal Rank (MRR). We empirically set the dimension of hidden states and all (POI and user) embedding size as 10 for all RNN-based techniques. We search the optimal temporal and spatial decay rate (α and β, respectively) on a log scale (see Section 4.3 for more details).

4.2 Location Prediction Performance

Table 2 shows the results on both Gowalla and Foursquare. In general, we observe that Flashback consistently and significantly outperforms all baseline techniques. In particular, compared to the best-performing baselines (spatiotemporal RNNs in most cases), Flashback shows an improvement of 27.7% and 15.9% in MRR, on Gowalla and Foursquare, respectively.

In addition, compared to the basic RNNs, Flashback consistently yields significant improvements of at least 27.5%, showing the effectiveness of leveraging past hidden states for next location prediction. Interestingly, we also observe a large variation on the performance of basic RNN, LSTM, and GRU (e.g., we see an MRR of 0.1507, 0.1144 and 0.0993 on the Gowalla dataset, respectively), showing their different capacities of modeling sparse user mobility traces. However, despite their different modeling capacities, Flashback implemented with RNN, LSTM, and GRU has a much smaller variation in terms of its performance (with an MRR of 0.1925, 0.1778 and 0.1731 on Gowalla, respectively). Such an observation shows that Flashback can boost the performance of any basic RNNs to a maximum extent, by fully benefiting from rich historical spatiotemporal contexts.

4.3 Impact of Spatiotemporal Decay Rates

In this experiment, we evaluate the impact of the temporal and spatial decay factors (α and β, respectively) on location prediction by varying α and β on a log scale. Figure 6 shows the results. On one hand, when increasing the spatial decay factor β, we observe that the prediction performance increases, and

[1] https://github.com/eXascaleInfolab/Flashback_code/

Method		Gowalla				Foursquare			
		Acc@1	Acc@5	Acc@10	MRR	Acc@1	Acc@5	Acc@10	MRR
User Preference based Methods	WRMF	0.0112	0.0260	0.0367	0.0178	0.0278	0.0619	0.0821	0.0427
	BPR	0.0131	0.0363	0.0539	0.0235	0.0315	0.0828	0.1143	0.0538
Feature-based Methods	MFT	0.0525	0.0948	0.1052	0.0717	0.1945	0.2692	0.2788	0.2285
	LBSN2Vec	0.0864	0.1186	0.1390	0.1032	0.2190	0.3955	0.4621	0.2781
Markov-Chain based Methods	FPMC	0.0479	0.1668	0.2411	0.1126	0.0753	0.2384	0.3348	0.1578
	PRME	0.0740	**0.2146**	**0.2899**	0.1503	0.0982	0.3167	0.4064	0.2040
	TribeFlow	0.0256	0.0723	0.1143	0.0583	0.0297	0.0832	0.1239	0.0645
Basic RNNs	RNN	0.0881	0.2140	0.2717	0.1507	0.1824	0.4334	0.5237	0.2984
	LSTM	0.0621	0.1637	0.2182	0.1144	0.1144	0.2949	0.3761	0.2018
	GRU	0.0528	0.1416	0.1915	0.0993	0.0606	0.1797	0.2574	0.1245
Spatiotemporal RNNs	DeepMove*	0.0625	0.1304	0.1594	0.0982	**0.2400**	0.4319	0.4742	0.3270
	STRNN	**0.0900**	0.2120	0.2730	**0.1508**	0.2290	0.4310	0.5050	0.3248
	STGN	0.0624	0.1586	0.2104	0.1125	0.2094	**0.4734**	**0.5470**	**0.3283**
	STGCN	0.0546	0.1440	0.1932	0.1017	0.1878	0.4502	0.5329	0.3062
Flashback	Flashback (RNN)	**0.1158**	**0.2754**	**0.3479**	**0.1925**	**0.2496**	**0.5399**	**0.6236**	**0.3805**
	Flashback (LSTM)	0.1024	0.2575	0.3317	0.1778	0.2398	0.5169	0.6014	0.3654
	Flashback (GRU)	0.0979	0.2526	0.3267	0.1731	0.2375	0.5154	0.6003	0.3631

Table 2: Location Prediction Performance on both Gowalla and Foursquare. The best-performing baselines and Flashback are highlighted. (*Experiments of DeepMove are conducted on 5,000 randomly sampled users, due to its poor efficiency where it takes more than one day per epoch using an NVIDIA V100 GPU for all users on both of our datasets.)

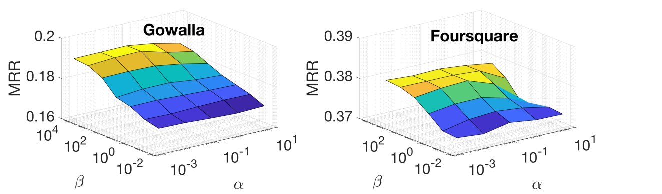


Figure 6: Impact of temporal and spatial decay rate (α and β, respectively)

then flattens out after $\beta = 100$ (on Foursquare) or $\beta = 1000$ (on Gowalla). This observation verifies the fact that the spatial distance is a very important factor defining context similarity for next location prediction problem; with a large β, the context similarity decreases fast with increasing spatial distances[2]. On the other hand, when decreasing temporal decay factor α, the prediction performance slightly increases, and achieves the best performance at $\alpha = 0.1$, implying a slow temporal decay over time which allows the hidden states at the periodicity peak to contribute more to location prediction. In other words, a low value of α indeed shows that the periodicity helps for predicting the next location.

4.4 How Far to Flash Back?

In this experiment, we study the impact of the number of involved past hidden states k (when flashing back) on the performance of location prediction. Figure 7 shows the results. We observe that performance increases with increasing k, as more hidden states can better help location prediction. The performance flattens out after $k \geq 20$, as further hidden states have little contribution due to the temporal decay, which corresponds to our previous discussion in Section 3.3.

[2]We compute spatial distance ΔD using the L2 distance (in kilometers) between the GPS coordinates of two POIs.

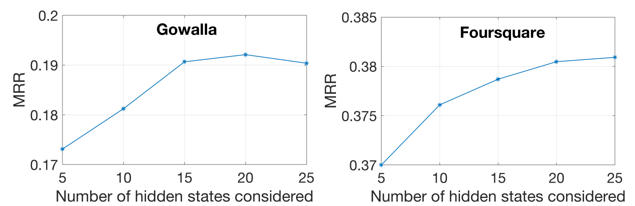


Figure 7: Impact of the number of involved past hidden states k when flashing back

5 Conclusion

In this paper, we propose Flashback, a general RNN architecture designed for modeling sparse user mobility traces by leveraging rich spatiotemporal contexts. Specifically, instead of implicitly considering context factors by adding spatiotemporal factors into the recurrent hidden state passing process of RNNs (as most existing techniques do), our solution explicitly uses the spatiotemporal contexts to search past hidden states with high predictive power (i.e., historical hidden states that share similar contexts as the current hidden state) for location prediction; subsequently, Flashback can directly benefit from rich spatiotemporal contexts encoded in user mobility traces. Our extensive evaluation compares Flashback against a sizable collection of state-of-the-art techniques on two real-world LBSN datasets. The results show that Flashback consistently and significantly outperforms state-of-the-art spatiotemporal RNNs by 15.9% to 27.6% when tackling the next location prediction task.

In future work, we plan to incorporate learnable spatiotemporal decay rates to fully automate our learning process.

Acknowledgements

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement 683253/GraphInt).

References

[Cheng *et al.*, 2013] Chen Cheng, Haiqin Yang, Michael R Lyu, and Irwin King. Where you like to go next: Successive point-of-interest recommendation. In *IJCAI*, 2013.

[Cho *et al.*, 2014] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv:1406.1078*, 2014.

[Feng *et al.*, 2015] Shanshan Feng, Xutao Li, Yifeng Zeng, Gao Cong, Yeow Meng Chee, and Quan Yuan. Personalized ranking metric embedding for next new poi recommendation. In *IJCAI*, 2015.

[Feng *et al.*, 2018] Jie Feng, Yong Li, Chao Zhang, Funing Sun, Fanchao Meng, Ang Guo, and Depeng Jin. Deepmove: Predicting human mobility with attentional recurrent networks. In *WWW*, pages 1459–1468, 2018.

[Figueiredo *et al.*, 2016] Flavio Figueiredo, Bruno Ribeiro, Jussara M Almeida, and Christos Faloutsos. Tribeflow: Mining & predicting user trajectories. In *WWW*, pages 695–706, 2016.

[Gao *et al.*, 2012] Huiji Gao, Jiliang Tang, and Huan Liu. Exploring social-historical ties on location-based social networks. In *ICWSM*, 2012.

[Gonzalez *et al.*, 2008] Marta C Gonzalez, Cesar A Hidalgo, and Albert-Laszlo Barabasi. Understanding individual human mobility patterns. *Nature*, 453(7196):779, 2008.

[Hidasi *et al.*, 2016] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. In *ICLR*, 2016.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[Hu *et al.*, 2008] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM*, pages 263–272. Ieee, 2008.

[Kong and Wu, 2018] Dejiang Kong and Fei Wu. Hst-lstm: A hierarchical spatial-temporal long-short term memory network for location prediction. In *IJCAI*, pages 2341–2347, 2018.

[Kurashima *et al.*, 2013] Takeshi Kurashima, Tomoharu Iwata, Takahide Hoshide, Noriko Takaya, and Ko Fujimura. Geo topic model: joint modeling of user's activity area and interests for location recommendation. In *WSDM*, pages 375–384. ACM, 2013.

[Liu *et al.*, 2016] Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. Predicting the next location: A recurrent model with spatial and temporal contexts. In *AAAI*, 2016.

[Mathew *et al.*, 2012] Wesley Mathew, Ruben Raposo, and Bruno Martins. Predicting future locations with hidden markov models. In *UbiComp*, pages 911–918. ACM, 2012.

[Mikolov *et al.*, 2010] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *INTERSPEECH*, 2010.

[Neil *et al.*, 2016] Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. Phased lstm: Accelerating recurrent network training for long or event-based sequences. In *NIPS*, pages 3882–3890, 2016.

[Noulas *et al.*, 2012] Anastasios Noulas, Salvatore Scellato, Neal Lathia, and Cecilia Mascolo. Mining user mobility features for next place prediction in location-based services. In *ICDM*, pages 1038–1043. IEEE, 2012.

[Rendle *et al.*, 2009] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI*, pages 452–461. AUAI Press, 2009.

[Rendle *et al.*, 2010] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *WWW*, pages 811–820. ACM, 2010.

[Xie *et al.*, 2016] Min Xie, Hongzhi Yin, Hao Wang, Fanjiang Xu, Weitong Chen, and Sen Wang. Learning graph-based poi embedding for location-based recommendation. In *CIKM*, pages 15–24. ACM, 2016.

[Yang *et al.*, 2013a] Dingqi Yang, Daqing Zhang, Zhiyong Yu, and Zhu Wang. A sentiment-enhanced personalized location recommendation system. In *HT*, 2013.

[Yang *et al.*, 2013b] Dingqi Yang, Daqing Zhang, Zhiyong Yu, and Zhiwen Yu. Fine-grained preference-aware location search leveraging crowdsourced digital footprints from lbsns. In *UbiComp*, pages 479–488, 2013.

[Yang *et al.*, 2015] Dingqi Yang, Daqing Zhang, Vincent W Zheng, and Zhiyong Yu. Modeling user activity preference by leveraging user spatial temporal characteristics in lbsns. *TSMC*, 45(1):129–142, 2015.

[Yang *et al.*, 2016] Dingqi Yang, Bin Li, and Philippe Cudré-Mauroux. Poisketch: Semantic place labeling over user activity streams. In *IJCAI*, pages 2697–2703, 2016.

[Yang *et al.*, 2019] Dingqi Yang, Bingqing Qu, Jie Yang, and Philippe Cudre-Mauroux. Revisiting user mobility and social relationships in lbsns: a hypergraph embedding approach. In *WWW*, pages 2147–2157. ACM, 2019.

[Zhang *et al.*, 2014] Yuyu Zhang, Hanjun Dai, Chang Xu, Jun Feng, Taifeng Wang, Jiang Bian, Bin Wang, and Tie-Yan Liu. Sequential click prediction for sponsored search with recurrent neural networks. In *AAAI*, 2014.

[Zhao *et al.*, 2019] Pengpeng Zhao, Haifeng Zhu, Yanchi Liu, Jiajie Xu, Zhixu Li, Fuzhen Zhuang, Victor S Sheng, and Xiaofang Zhou. Where to go next: A spatio-temporal gated network for next poi recommendation. In *AAAI*, volume 33, pages 5877–5884, 2019.

[Zhu *et al.*, 2017] Yu Zhu, Hao Li, Yikang Liao, Beidou Wang, Ziyu Guan, Haifeng Liu, and Deng Cai. What to do next: Modeling user behaviors by time-lstm. In *IJCAI*, pages 3602–3608, 2017.