TPM: A Temporal Personalized Model for Spatial Item Recommendation

WEIQING WANG, The University of Queensland, Australia
HONGZHI YIN*, The University of Queensland, Australia
XINGZHONG DU, The University of Queensland, Australia
QUOC VIET HUNG NGUYEN, Griffith University, Australia
XIAOFANG ZHOU, The University of Queensland, Australia

With the rapid development of location-based social networks (LBSNs), spatial item recommendation has become an important way of helping users discover interesting locations to increase their engagement with location-based services. The availability of spatial, temporal and social information in LBSNs offers an unprecedented opportunity to enhance the spatial item recommendation. Many previous work studied spatial and social influences on spatial item recommendation in LBSNs. Due to the strong correlations between a user’s check-in time and the corresponding check-in location, which include the sequential influence and temporal cyclic effect, it is essential for spatial item recommender system to exploit the temporal effect to improve the recommendation accuracy. Leveraging temporal information in spatial item recommendation is, however, very challenging, considering 1) when integrating sequential influences, users’ check-in data in LBSNs has a low sampling rate in both space and time, which renders existing location prediction techniques on GPS trajectories ineffective and the prediction space is extremely large, with millions of distinct locations as the next prediction target, which impedes the application of classical Markov chain models; 2) there are various temporal cyclic patterns (i.e., daily, weekly and monthly) in LBSNs, but existing work is limited to one specific pattern; and 3) there is no existing framework that unifies users’ personal interests, temporal cyclic patterns and the sequential influence of recently visited locations in a principled manner.

In light of the above challenges, we propose a Temporal Personalized Model (TPM) which introduces a novel latent variable topic-region to model and fuse sequential influence, cyclic patterns with personal interests in the latent and exponential space. The advantages of modeling the temporal effect at the topic-region level include a significantly reduced prediction space, an effective alleviation of data sparsity and a direct expression of the semantic meaning of users’ spatial activities. Moreover, we introduce two methods to model the effect of various cyclic patterns. The first method is a time indexing scheme which encodes the effect of various cyclic patterns into a binary code. However, the indexing scheme faces the data sparsity problem in each time slice. To deal with this data sparsity problem, the second method slices the time according to each cyclic pattern separately and explores these patterns in a joint additive model.

Furthermore, we design an asymmetric Locality Sensitive Hashing (ALSH) technique to speed up the online top-k recommendation process by extending the traditional LSH. We evaluate the performance of TPM on two real datasets and one large-scale synthetic dataset. The performance of TPM in recommending cold-start items

*Corresponding author

Authors’ addresses: Weiqing Wang, The University of Queensland, Australia, wq.wu@gmail.com; Hongzhi Yin, The University of Queensland, Australia, h.yin1@uq.edu.au; Xingzhong Du, The University of Queensland, Australia, domainxz@gmail.com; Quoc Viet Hung Nguyen, Griffith University, Australia, quocviethung1@gmail.com; Xiaofang Zhou, The University of Queensland, Australia, xzf@itee.uq.edu.au.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2018 Association for Computing Machinery.

XXX-X-XXXX/2018/4-A RT $15.00
https://doi.org/10.1145/nnnnnn.nnnnnn

, Vol. 1, No. 1, Article . Publication date: April 2018.
is also evaluated. The results demonstrate a significant improvement in TPM’s ability to recommend spatial items, in terms of both effectiveness and efficiency, compared with the state-of-the-art methods.

CCS Concepts: • Information systems → Spatial-temporal systems; Data mining; Retrieval models and ranking;

Additional Key Words and Phrases: POI; Location-based Service; Online Learning; Spatial-Temporal Modeling

ACM Reference Format:

1 INTRODUCTION

The rapid development of Web 2.0, location acquisition and wireless communication technologies have fostered a number of location-based social networks (LBSNs), such as Foursquare, Gowalla, Brightkite and Loopt, where users can check in at different venues and share life experience in the physical world via mobile devices [1]. A personalized location recommendation service, which encourages users to explore new locations [9], is an essential function of LBSNs. Therefore, developing personalized recommender systems for LBSNs to provide users with spatial items (e.g., a venue or a POI) has recently attracted increased research attention [13, 36, 37].

Existing research on personalized spatial item recommendation mainly explores the geographic influence to improve the recommendation accuracy, based on the observation that the geographic proximity between spatial items affect users’ check-in locations [13, 15]. Recently, there are works that further integrate social influence in LBSNs to recommend items as common interests shared by social friends [1]. In terms of the temporal effect of user check-in activities in LBSNs, many recent studies have shown that both temporal cyclic and sequential patterns of check-ins significantly affect users’ activities [6, 10, 22, 27]. Temporal cyclic patterns refer to that users prefer different spatial items at different time [43]. On the other hand, human movement also exhibits sequential patterns. These sequential patterns result from different factors, such as time in one day (e.g., people tend to go to restaurants at dinner time and then relax in cinemas or bars at night [41]), geographical proximity (e.g., tourists often sequentially visit London Eye, Big Ben and Downing Street [38]) and the coherence between human preference and the type of places (e.g., people usually check in at a stadium before a restaurant instead of the reverse way because it is not healthy to exercise right after a meal [11]). Therefore, exploiting both the temporal cyclic patterns and sequential influence of user check-in behaviors is important to improve personalized location recommendation in LBSNs.

Nevertheless, Leveraging temporal information for spatial item recommendation is highly challenging, mainly due to the following problems:

Low-sampling rate. There are a number of studies that predicate locations on GPS trajectories [23, 46]. At first glance, these approaches can be directly applied to integrating the sequential patterns in LBSN data, since both the GPS and LBSN data contain location and time information. However, analysis [4] of the check-in records collected from Gowalla, a popular LBSN, shows that LBSN data has a low sampling rate in both space and time, compared to GPS trajectories. According to the analysis, only 10% of users have more than 58 check-in records over a 12-month period, representing a low check-in frequency over time. In addition, 40% of all consecutive check-ins have a spatial distance larger than 1 kilometer, much longer than the gap in GPS trajectories which is typically 5-10 meters [46]. Thus, it is difficult to model the dependency between two check-in locations in LBSNs using the location prediction techniques on GPS trajectories.

Huge prediction space. Sequential recommendation methods have been proposed in the literatures [3, 42, 47], most of which are based on Markov chains. Suppose there are a collection of V
spatial items and the next item depends on the previous \( n \) items. The sequential recommendation methods then need to estimate \( |V|^{n+1} \) free parameters in the \( n \)th order Markov chain model, which is extremely computational-expensive. To reduce the size of the prediction space, most related studies [3, 47] exploit sequential influence using a first-order Markov chain, which considers only the last one in a sequence of locations visited by a user to recommend a new location for her. Although the parameter space can be decreased to \( |V|^2 \), it may still be huge considering that \( V \) is usually a large number in LBSNs. Hence, we aim to develop a new method to incorporate the influence from all recently visited locations, rather than just the last one, to make location recommendations within a small prediction space.

Various temporal cyclic patterns. There are various temporal cyclic patterns affecting users’ check-in behaviors [10]. Most existing work only leverage one temporal cyclic pattern [6, 22, 26, 28]. To illustrate the various temporal cyclic patterns in LBSNs, we analyze the check-in distributions in time slices divided by different granularities on Foursquare. Following [10], we assume that there are three main cyclic patterns affecting users’ check-in behaviors: daily (hour in one day), weekly (day of week) and yearly (month in one year). There are 36 categories on Foursquare and we show the check-in distributions of 8 selected categories for each cyclic pattern. From Figure 1(a), we can see that most categories exhibit a similar daily pattern - the check-ins start to decrease since midnight and reach the minimum at 10 am. The check-in activities start to rise at 10 am and peak in the afternoon or evening. But the number of check-ins associated with some categories, i.e., nightlife spot, peak in the midnight. Figure 1(b) tells that people tend to visit some places (such as gym, college, office and so on) in the weekdays while prefer other places (i.e., clothing stores and performing arts venues) in the weekends. Monthly pattern refers to that people prefer to visit different locations in different months. For example, people prefer to go skiing in the winner months while go to beach in the summer months. For example, people prefer to go skiing in the winner months while go to beach in the summer months. For example, people prefer to go skiing in the winner months while go to beach in the summer months.

Unifying personalization, cyclic effect and sequential influence. On one hand, most existing spatial item recommendation methods focus on personalization [1, 6, 9] and make recommendations according to users’ personal interests, but neglect the temporal effect. On the other hand, existing temporal recommendation methods either only explore the cyclic effect or only exploit the sequential influence. Moreover, existing sequential recommendation methods, such as Markov chain based approaches, capture sequential patterns by assuming equivalent transition probabilities between items for all users, and ignore personalization. One recent work has been proposed in [28] integrating the three factors. However, [28] overlooks personalization and weight in integrating different factors. On one hand, CIKM 16 captures sequential patterns through a shared POI-POI graph embedding for all users. In this way, it assumes equivalent transition probabilities between items for all users, and ignores personalization. On the other hand, CIKM 16 combines the sequential effect, cyclic patterns and personalization through a simply linear combination without any weight scheme. Therefore, we
aim to develop a recommendation method to combine both personalization and temporal influence in a unified and principled manner.

In light of the aforementioned challenges, we propose a **Temporal Personalized Model** for spatial item recommendation, called TPM, which seamlessly fuses the temporal influence and the personal interests of individual users in a principled way. Technically, TPM is a latent class probabilistic generative model designed to mimic users’ decision-making process for choosing spatial items. We model personal interests and temporal influence based on the latent variable *topic-region* in TPM. Wang et al. model each topic-region with a semantic topic and a soft cluster of spatial items in [27]. The soft clusters of spatial items are obtained by mining the items’ co-occurrence. A topic-region *z* in TPM corresponds to a semantic topic and a geographical region (i.e., a soft cluster of geographical locations of spatial items). Modeling topic-region with geographical locations instead of spatial item IDs enables TPM to recommend cold start spatial items.

By introducing the latent factor *topic-region*, TPM effectively overcomes the issue of low-sampling rates. Specifically, TPM addresses the sparsity of LBSN data by considering the hidden variable *topic-region*, which groups spatial items with similar semantic meanings and geographical locations, rather than focusing on the fine granularity of data such as consecutive points in GPS trajectories.

Our proposed TPM is able to reduce the prediction space effectively when modeling the sequential influence. In particular, for each spatial item *v*, we learn a distribution $\theta^{seq}_{v}$ over a set of topic-regions where each component $\theta^{seq}_{v, z}$ represents the probability of visiting the topic-region *z* after visiting *v*. An obvious advantage of predicting the topic-region of a user’s activity at the next step is a significantly reduced prediction and model parameter space, because the number of topic-regions is much smaller than the number of spatial items. Additionally, to capture the influence from high order items, TPM adds the influence of the previously visited items in the exponential space to avoid the inference of mixture weights for each visited item, inspired by the Sparse Additive Generative model (SAGE) [8]. In this way, TPM accurately captures the influence from more items previously visited by the target user and also reduces the exponential complexity $|V|^{n+1}$ of the classic *n*th order Markov Chain into linear complexity $|V| \times K$ (where *K* is the number of topic-regions).

To integrate various temporal cyclic patterns, we first introduce a three-slice time indexing scheme. The preference variance exists in three scales following [10]: hours of a day, weekdays/weekends and different months of a year. The time indexing scheme smoothly encodes a standard time stamp to a particular time id capturing the three cyclic patterns. This method encodes the three patterns out of TPM and the new time ids are the input of TPM. Assume that all the time stamps are divided into $N_y$, $N_w$ and $N_d$ time slices in terms of month, weekday and hour respectively, then the data is divided into $N_y \times N_w \times N_d$ slices. In this way, the data in each time slice is extreme sparse. To combat this issue, we introduce another method to combine the three patterns inside TPM. This method divides the time stamps according to the three patterns separately. In this way, each time stamp has three time ids with respect to three cyclic patterns respectively. Then TPM combines these patterns with the SAGE model.

To unify personal interests, temporal cyclic effect and sequential influence in a principled way, traditional mixture models, such as LCA-LDA [35], combine multiple facets (e.g., personal interests and temporal effect) by introducing additional latent variables that act as “switches”, to control which facet is currently active. In reality, it is computationally expensive and difficult to learn these variables accurately, given sparse datasets. Again, TPM follows the SAGE model [8] to add the
effect of personal interests, cyclic effect and sequential influence in the exponential space to avoid the inference of latent “switching” variables, to improve both robustness and predictive accuracy.

To support real-time recommendation scenario, we further design an asymmetric Locality Sensitive Hashing (ALSH), extending the classical LSH technique, to significantly reduce the search space and produce top-\(k\) recommendations without examining all available spatial items.

Note that, we presented our preliminary study of sequential personalized spatial item recommendation and proposed a recommender model SPORE in [27]. In this article, we extend [27] with an in-depth analysis. Specifically, this article makes the following new contributions: 1) we extend SPORE to TPM by exploiting and integrating the temporal cyclic effect of user visiting behaviors; 2) we introduce two novel methods to model the various cyclic patterns in a unified way; 3) we optimize the way of obtaining topic-regions to support the recommendation for cold start items; 4) we redo the experiment and conduct a more comprehensive analysis in terms of both recommendation effectiveness and efficiency; 5) we provide a more comprehensive review of the related work.

2 RELATED WORK
There are two main lines of research in spatial item recommendation. The first one focuses on GPS trajectory data. GPS trajectory data usually consists of a small number of users, but has dense location records [2, 45]. The other line of research is conducted on LBSN data, which has a low sampling rate in both space and time compared to GPS trajectories [4, 46]. Many recent studies have shown that there is a strong correlation between user check-in activities and geographical distance as well as social connections [13]. Consequently, most current spatial item recommendation methods focus on leveraging geographical and social influences to improve recommendation accuracy.

Recently, the temporal effect of user check-in activities in LBSNs has been noted in spatial item recommendation [10, 31, 32, 34]. Most existing work focus on leveraging the temporal cyclic patterns of user check-ins to capture the temporal non-uniformness and temporal consecutiveness in recommending spatial items [6, 10, 22, 26]. Although it has been shown that there are various temporal patterns affecting users’ check-in behaviors [10], most research only leverage one temporal pattern. For example, Wang et al. explore the effect of various cyclic patterns but only integrate the daily pattern which has the biggest effect into the model in [26]. Compared with integrating temporal cyclic patterns, leveraging sequential patterns for spatial item recommendation has not been well-studied. However, it has been shown in multiple studies that human movement in LBSNs clearly demonstrates sequential patterns [11, 38, 41].

Existing works on sequential recommendation mostly utilize the Markov chain property to predict the next check-ins. Cheng et al. [3] exploited sequential influence using the first-order Markov chain that only considers the last location in a user’s visiting sequence to recommend a new location for the user. However, in reality, the new location may not only rely on the latest location but also earlier ones visited by the user [42]. Zhang et al. predicted the next location by adding the influence of the earlier visited locations with an additive Markov chain. They manually set a decay rate parameter for previous locations based on the assumption that locations with recent check-in timestamps usually have stronger influence than those with old timestamps. To adapt to the sparse data in LBSNs, Ye et al. [4] proposed modeling the sequential patterns of spatial items at the category level using a hidden Markov model (HMM). The abstract states in HMM can model sparse LBSN data well as the hidden states capture essential behavioral patterns of LBSN users. The accuracy of this method depends highly on the category information.

Our work in this paper distinguishes itself from previous research in several aspects. Firstly, to the best of our knowledge, this is the first effort that automatically integrates sequential effect, cyclic patterns and personalization in a principled and personalized way. Secondly, most existing work leverage only one cyclic pattern while we propose two novel methods to integrate the various cyclic
patterns seamlessly. Thirdly, although existing research [42] has also exploited the influence of more than one previously visited locations using an additive Markov chain method, the weights for each visited item in a sequence need to be set manually. In contrast, our proposed TPM model adopts the sparse additive technique to add the influence of all visited items in the exponential space, which avoids the inference of item weights. Last, we introduce a novel latent variable topic-region to model both personal interests and temporal influence in the exponential space. The discovered topic-regions cluster both content-similar and geographically close items together. By introducing the latent factor topic-region, TPM effectively overcomes the challenge brought by low-sampling rate and reduces the prediction space.

### 3 Preliminaries and Problem Formulation

In this section, we formally define the data in LBSNs that we are interested in and formulate the problem accordingly. For ease of presentation, Table 1 lists the notations.

**Definition 3.1. (Spatial Item)** A spatial item is an item associated with a geographical location (e.g., a restaurant or a cinema).

In our model, a spatial item has three attributes: identifier, location and contents. We use \( v \) to represent a spatial item identifier, \( l_i \) to denote its corresponding location and \( W_v \) to represent the set of words describing the semantic information of the item (e.g., tags and categories). POI is a kind of spatial item.

**Definition 3.2. (Sequence)** A sequence of user \( u \), denoted by \( S_u = \{(v_1, t_1), (v_2, t_2), \ldots, (v_n, t_n)\} \), consists of an ordered list of elements, where each element \((v_i, t_i)\) indicates that user \( u \) visited spatial item \( v_i \) at time \( t_i \) (1 ≤ \( i \) ≤ \( n \) and \( t_1 \leq t_2 \leq \ldots \leq t_n \)).

**Definition 3.3. (Predecessor, Successor)** Given a sequence \( S_u = \{(v_1, t_1), (v_2, t_2), \ldots, (v_n, t_n)\} \) and a time period threshold \( \Delta T \), if \( v_i \) and \( v_j \) are two items in this sequence and \( 0 < t_i - t_j \leq \Delta T \), we say \( v_j \) is a **predecessor** of \( v_i \). Conversely, \( v_i \) is a **successor** of \( v_j \).

**Definition 3.4. (Predecessor Set)** Given a target user \( u \) and time \( t \), the predecessor set, denoted as \( P_{u,t} = \{v_i \mid v_i \in S_u, 0 < t - t_i \leq \Delta T\} \), is a set of spatial items visited before \( t \) in the given time threshold \( \Delta T \).

Following the previous works [41, 42], we assume that only the spatial items in the predecessor set have sequential influence on a user’s decision-making. That is, if the temporal interval between
two spatial items is greater than the specified threshold $\Delta T$, it is assumed that there is no influence between the two items. We will study the impact of $\Delta T$ on the quality of spatial item recommendation in Section VI.

**Definition 3.5.** *(User Activity)* A user activity is a four tuple $(u, v, t, P_{u,t})$, which indicates that the user $u$ visits the spatial item $v$ at time $t$. $P_{u,t}$ is the predecessor set of spatial items that user $u$ has visited before $t$.

**Definition 3.6.** *(User Profile)* For each user $u$, we create a user profile $D_u$, which contains a set of user activities associated with $u$. Given a set of users $U$, the dataset $D$ used in our model is a collection of user profiles, $D = \{D_u : u \in U\}$.

Then, given a dataset $D$ as the union of a collection of user profiles, we aim to provide spatial item recommendations for users, stated as follows.

**Problem 1.** *(Spatial Item Recommendation)* Given a user activity dataset $D$ and a querying user $u$ at time $t$ (i.e., the query is $q = (u, t)$), our goal is to recommend a list of spatial items that $u$ would be interested in.

In our research, we aim to integrate personal interests, cyclic effect and sequential influence into a unified framework. In particular, there are four main tasks involved in our problem.

**Task 1: Extracting users’ personal interests.** This task models the users’ personal interests.

**Task 2: Extracting sequential influence.** This task models the influence of visited spatial items in a sequence.

**Task 3: Extracting temporal cyclic effect.** This task models temporal cyclic patterns.

**Task 4: Fusing users’ personal interests, sequential influence and cyclic effect into a unified framework.**

All four tasks are very challenging in the context of LBSN data. In Task 1, the user-item matrix is very sparse in LBSNs, which makes it difficult for traditional recommendation models (e.g., matrix factorization or collaboration filtering) to accurately infer users’ interests from the data. As analyzed in Section 1, Task 2 faces the severe challenges of low-sampling rate and huge prediction space, which render the classical $n$th order Markov Chain inefficient because its complexity increases exponentially w.r.t. $n$ (corresponding to $|P_{u,t}|$) [42]. In Task 3, there are various cyclic patterns and we aim to model these patterns in a unified way. In Task 4, no framework exists that simultaneously integrates the three components into a unified model. Zhang et al. combine the sequential influence, geographical influence and social influence in [42]. However, they fuse the three components by simply multiplying them together while we try to unify them in a principled manner rather than an ad-hoc way.

**4 THE TPM MODEL**

In this section, we first present some preliminaries about the SAGE model [8], and then describe our temporal personalized recommendation model based on it.

**4.1 Preliminaries about SAGE**

Our model is inspired by the Sparse Additive Generative Model (SAGE) [8], which is an effective generative model without explicit “switching” variables. The basic idea of the model is that, if a variable is affected by several factors, it can be generated by the mixture of these factors without any explicit indicator variable. The key difference between SAGE and traditional mixture models is that the mixture occurs in terms of natural parameters of the exponential family instead of distributions. Such a model is robust given limited training data as it does not have to infer a complex indicator variable distinguishing the set of factors.
To provide a clearer explanation of SAGE, we take its application to our **Task 4** as an example. Given a query \((u, t)\), we retrieve the predecessor set \(P\) which contains items visited by \(u\) before \(t\). A traditional probabilistic mixture generative model combines the three factors, \(u\)'s personal interests \(\theta_{user}\), the sequential influence of all the predecessors \(\theta_{seq}\) and the temporal cyclic effect \(\theta_{time}\), through a linear combination as Equation 1.

\[
P(v | \theta_{user}^u, \theta_{seq}^p, \theta_{time}^t) = \lambda_u P(v | \theta_{user}^u) + \lambda_u' P(v | \theta_{seq}^p) + (1 - \lambda_u - \lambda_u') P(v | \theta_{time}^t)
\]

where \(\lambda_u\) and \(\lambda_u'\) are the personalized mixture weights (i.e., the “switching” variables) that need to be inferred for each user. Obviously, it is very difficult to infer the variables accurately when the training data for the individual user is sparse. By contrast, SAGE combines the three generative facets through simple addition in an exponential space as illustrated in Equation 2. Note that SAGE avoids the need for computing latent switching variables.

\[
P(v | \theta_{user}^u, \theta_{seq}^p, \theta_{time}^t) = \frac{e^{\theta_{user}^u + \theta_{seq}^p + \theta_{time}^t}}{\sum_v e^{\theta_{user}^u + \theta_{seq}^p + \theta_{time}^t}}
\]

### 4.2 Model Structure

**Overview of TPM.** TPM is a probabilistic generative model that aims to mimic the process of human decision making when visiting spatial items. It assumes that a user \(u\)’s decision-making at time \(t\) is influenced by three factors: 1) her personal interests \(\theta_{user}^u\); 2) the influence of the items visited before \(t\), \(\theta_{seq}^p = \{\theta_{seq}^{p_1}, \theta_{seq}^{p_2}, \ldots, \theta_{seq}^{p_l}\}\); 3) the temporal cyclic influence \(\theta_{time}^t\). Figure 2 shows the graphical representation of TPM, and Table 2 introduces the notations of model parameters. Our input data (i.e., users’ activity profiles) are modeled as observed random variables in our model, shown as shaded circles (e.g., \(t, u, l\) etc.) in Figure 2. Because a spatial item has both semantic and geographical attributes, we introduce a joint latent variable **topic-region** which corresponds to both a semantic topic (i.e., a soft cluster of words) and a geographical region (i.e., a soft cluster of locations). All three components (i.e., personal interests, sequential influence and the temporal cyclic effect) are modeled as a distribution over a set of topic-regions and influence \(u\)’s decision-making by generating a topic-region. 

---

![Graphical Representation of Our Model](image-url)
Table 2. Notations of Model Parameters

<table>
<thead>
<tr>
<th>Variable</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K$</td>
<td>the number of topic-regions</td>
</tr>
<tr>
<td>$z_{u,i}$</td>
<td>the topic-region assigned to spatial item $v_{u,i}$</td>
</tr>
<tr>
<td>$\theta^v$</td>
<td>the topic-region distribution of the background</td>
</tr>
<tr>
<td>$\theta^u_{\text{user}}$</td>
<td>the topic-region distribution, denoting the intrinsic interest of user $u$</td>
</tr>
<tr>
<td>$\theta^v_{p_j,u}$</td>
<td>the topic-region distribution of $v^j$th spatial item in $p_j$</td>
</tr>
<tr>
<td>$\phi^z_{\text{topic}}$</td>
<td>content word distribution of topic-region $z$</td>
</tr>
<tr>
<td>$\mu_z$</td>
<td>the mean location of the topic-region $z$</td>
</tr>
<tr>
<td>$\Sigma_z$</td>
<td>the covariance matrix of the topic-region $z$</td>
</tr>
</tbody>
</table>

The set of topic-regions are obtained by simultaneously mining both the spatial patterns of spatial items and their content information (e.g., tags and categories). Different from our preliminary work in [27], TPM mines the topic-regions from location information instead of the co-occurrence of items. In [27], if several spatial items are frequently visited by same users, they are more likely to be put into the same topic-regions. For cold-start items, it is difficult to mine the topic-regions from the co-occurrence of items as they usually have been visited by few users. Different from [27], TPM assumes that if two items have close location, they are more likely to be put into the same topic-regions. In this way, TPM is able to model the cold-start items without visiting information as long as their location information is available. Thus, introduction of the latent variable topic-region is very helpful in alleviating the issues caused by data sparsity and low-sampling rates. It also contributes to reducing the prediction and parameter space, as the number of topic-regions is much smaller than the number of spatial items.

**Personalization Component.** Inspired by early work on user interest modeling [12], TPM adopts latent topic-regions to characterize users’ interests in terms of both semantic and geographical aspects. Specifically, we infer an individual user’s interest distribution over a set of topic-regions according to the spatial patterns of her visited spatial items and their associated contents, denoted as $\theta^u_{\text{user}}$. To capture both the semantic and spatial information, a topic-region $z$ in TPM is associated with a word distribution $\phi^z_{\text{topic}}$ and a distribution over $K$ regions. Following literatures [14, 34], we assume a Gaussian distribution for each region $z$, and the check-in location is characterized by $l \sim N(\mu_z, \Sigma_z)$.

Note that in the traditional topic models such as LDA, a document contains a mixture of topics, and each word has a hidden topic label. This is reasonable for long documents. However, the “document” $W_v$ associated with a check-in activity is usually short, and is likely to only contain a single topic [44]. Thus in TPM, all the words in $W_v$ are assigned with a single topic $z$, and they are generated from the same word distribution $\phi^z_{\text{topic}}$.

**Sequential Component.** Given the $i^{th}$ check-in activity of user $u$, let its timestamp be $t_{u,i}$, we model the influence of each spatial item in the predecessor set $P_{u,t_{u,i}}$ denoted as $\theta^{seq}_{p_1}, \theta^{seq}_{p_2}, \ldots, \theta^{seq}_{p_{|P_{u,t_{u,i}}|}}$. To reduce the exponential complexity of classical $n$th-order Markov Chain to the linear complexity, an intuitive solution is to add the influence of each predecessor, inspired by the idea of $n$th additive Markov Chain in [41, 42]. However, using the $n$th additive Markov Chain has the following limitations: 1) it is an ad-hoc method that requires manual setting of weighting schemes with a decay rate parameter; 2) it ignores the difference between various users by using one common decay rate parameter for all users. In reality, different users tend to have different check-in frequencies and time intervals. Thus, the influence of $n$th items in the predecessor sets of each user are different. That is, the weights of predecessors should be personalized. One natural solution to overcome the
two limitations is to learn personalized weights for each user using a traditional mixture model as in Equation 3. However, training data is often sparse in LBSNs, especially for each user. It is hard to infer the weighting variables accurately with limited training data. To handle this sparsity problem, inspired by SAGE, TPM transforms the traditional mixture model into a mixture occurring in terms of natural parameters of the exponential family instead of distributions to avoid computing a weighting scheme for each user, as in Equation 4. In this way, TPM also reduces the exponential complexity \(|V|^n+1\) of classical nth order Markov Chain into \([V] \times K (n \text{ corresponds to } |P_{u,t}| \text{ in TPM}).

\[
P(v|\theta_{seq}^{seq}) = \sum_{j} \lambda_{u,j} P(v|\theta_{seq}^{seq}) \lambda_{u,j} = 1 (\lambda_{u,j} \geq 0) \tag{3}
\]

\[
P(v|\theta_{seq}^{seq}) = P(v|\theta_{per} + \theta_{week} + \theta_{day}) = \frac{\exp(\sum_{j} \lambda_{u,j} \theta_{seq}^{seq})}{\sum_{v} \exp(\sum_{j} \lambda_{u,j} \theta_{seq}^{seq})} \tag{4}
\]

**Temporal Cyclic Component.** Recent studies show that users’ check-ins exhibit strong temporal cyclic patterns [5, 10, 29, 44]. For example, users tend to check in around the office in weekdays; users prefer restaurants at noon and bars at night. The cyclic patterns exist in different scales: hours of a day (i.e. daily), different days of a week (i.e. weekly), different months of a year (i.e. monthly), various seasons of a year (i.e. seasonly) and so on [10]. However, most existing work only highlight the modeling of single cyclic pattern [10, 31, 34]. Following [10], we assume that there are three main cyclic patterns affecting users’ check-in behaviors: yearly (month in one year), weekly (day of week) and daily (hour in one day). To integrate the effect of the three cyclic patterns, we introduce a three-slice time indexing scheme [43]. First, a time stamp is divided into three slices in terms of month, day of week, and hour. Then, we use 4 bits to represent the month information, 3 bits to denote the weekday and 5 bits to define the hour in one day. Finally, we convert the binary code into a unique decimal digit as the time ID. In this way, we can obtain \(T = 12 \times 7 \times 24 = 2016\) time slices. Figure 3 demonstrates the procedure of encoding an exemplary time stamp, “2016-07-05 20:15:50”. However, the check-in data in each time slice becomes extremely sparse, this makes the learning of users’ preferences in each time slice \(t\), denoted by \(\theta_{time}^{t}\), inaccurate.

![Fig. 3. Time Indexing Scheme Demonstration](image)

To combat the data sparsity problem of the three-slice time indexing scheme, we propose another method to encode the three patterns. This method indexes each time stamp \(t\) with three patterns separately. As a result, each time stamp \(t\) is transformed into three indexes \(t_{month}, t_{weekday}\) and \(t_{hour}\). Based on these indexes, TPM learns \(\theta_{year}, \theta_{week}\) and \(\theta_{day}\) instead of \(\theta_{time}^{t}\) in Figure 2 separately. For \(\theta_{year}, \theta_{week}\) and \(\theta_{day}\), time stamps in the data set are divided into 12, 7 and 24 slices respectively. Then, TPM combines the three patterns in the sparse additive generative model as Equation 5. In this way, we transfer the time slicing from product space into additive space. As we can see, the time
we sample latent topic-region assignments by fixing all of the other parameters using Gibbs sampling.

\[ \alpha_{u, t, z} = \frac{\exp(q_{t, z}^{time} + \theta_{u, z}^{user} + \sum_{i=1}^{\|P_{u, i}\|} \theta_{i, z}^{seq})}{\sum_{z} \exp(q_{t, z}^{time} + \theta_{u, z}^{user} + \sum_{i=1}^{\|P_{u, i}\|} \theta_{i, z}^{seq})} \]

\[ \beta_{z, w} = \frac{\exp(\phi_{w}^{0} + \phi_{z, w}^{topic})}{\sum_{w} \exp(\phi_{w}^{0} + \phi_{z, w}^{topic})}, \]

\[ \gamma_{z, l} = \frac{1}{2\pi \sqrt{\Sigma_{z}}} \exp\left(-\frac{(I - \mu_{z})^T \Sigma_{z}^{-1} (I - \mu_{z})}{2}\right) \]

\[ P(z, w, l | \varnothing, u, t, P) = P(z | u, t, P, \theta^{time}, \theta^{user}, \theta^{seq}) P(w | z, \phi^{0}, \phi^{topic}) P(l | \mu, \Sigma) \]

\[ = \prod_{u=1}^{n} \sum_{l=1}^{\|D_{u}\|} \sum_{z=1}^{\|W_{u, t}\|} \beta_{z, w_{u, t, i}, n} \beta_{u, t, z_{u, t, i}} \gamma_{z_{u, t, i}, l_{u, t}} \]

indexing scheme combines the three patterns outside the model while this method combines these patterns inside the model.

\[ P(z | \theta_{u, t, i}^{time}) = P(z | \theta_{u, t, i}^{time} + \theta_{u, t, i}^{user} + \theta_{u, t, i}^{seq}) \]

**4.3 Generative Process of TPM**

The generative process of the TPM model for a user check-in activity is as follows.

- Draw a topic-region index \( z_{u, i} \)
  \( z_{u, i} \sim P(z_{u, i} | P_{u, t, i}, \theta^{time}, \theta^{user}, \theta^{seq}) \)
- For each content word \( w_{u, t, i, n} \) in \( W_{u, t, i} \), draw \( w_{u, t, i, n} \sim P(w_{u, t, i, n} | \phi^{0}, z_{u, t, i}, \phi^{topic}) \)
- Draw a location \( l_{u, i} \)
  \( l_{u, i} \sim P(l_{u, i} | z_{u, t, i}, \mu, \Sigma) \)

For each user activity, TPM first chooses the topic-region this activity is about. To generate the topic-region index \( z \), we utilize a multinomial model as expressed in Equation 6.

\[ P(z_{u, i} | P_{u, t, i}, \theta^{time}, \theta^{user}, \theta^{seq}) = P(z_{u, i} | \theta_{u, t, i}^{time} + \theta_{u, t, i}^{user} + \theta_{u, t, i}^{seq}) \]

where \( \theta_{P_{u, t, i}}^{seq} \) is the sum of the influences of all the visited spatial items in \( P_{u, t, i} \). It can be expanded as \( \sum_{n=1}^{\|P_{u, t, i}\|} \theta_{P_{u, t, i, n}}^{seq} \). Once the topic-region \( z \) is generated, the location of spatial item \( v \) and the associated content words are generated as expressed in Equations 7 and 8, respectively.

\[ P(l_{u, i} | z_{u, i}, \mu, \Sigma) = \frac{1}{2\pi \sqrt{\Sigma_{z}}} \exp\left(-\frac{(l_{u, i} - \mu_{z_{u, i}})^T \Sigma_{z_{u, i}}^{-1} (l_{u, i} - \mu_{z_{u, i}})}{2}\right) \]

\[ P(w_{u, t, i, n} | \phi^{0}, z_{u, t, i}, \phi^{topic}) = P(w_{u, t, i, n} | \phi^{0} + \phi_{z_{u, t, i}}^{topic}) \]

**4.4 Model Inference**

Our goal with model inference is to learn the parameters that maximize the marginal log-likelihood of the observed random variables \( w \) and \( l \). Marginalization is performed with respect to the latent random variable \( z \). However, it is difficult to achieve maximization directly. Therefore, we apply a mixture between EM and a Monte Carlo sampler, called the Gibbs EM algorithm [24], to maximize the complete data likelihood in Equation 10, where \( \varnothing \) is the set of all the parameters. In the E-step, we sample latent topic-region assignments by fixing all of the other parameters using Gibbs sampling. In the M-step, we optimize the model parameters \( \varnothing \) by fixing all topic-region assignments. The two steps are iterated until convergence.

More specifically, we iteratively draw the latent topic-region \( z \) for all check-in activities in the E-step. According to the Gibbs Sampling, when sampling \( z_{u, i} \) as expressed in Equation 11, we assume
all other variables are fixed. \( z_{-u,i} \) represents the topic-region assignments for all user activities except the \( i^{th} \) activity for user \( u \).

\[
P(z_{u,i}|z_{-u,i}, w, l, u, t, P, \odot) \propto a_{u_i,t_i,z_{u_i}} \times \prod_{n=1}^{[w_{z_{u,i}}]} \beta_{z_{u,i}w_{u,i,n}} \times \gamma_{z_{u,i}t_{u,i}}
\] (11)

In the M-step, we optimize the parameters \( \odot \) to maximize the log likelihood of the objective function with all topic-region assignments fixed. To update the parameters, we use the gradient descent learning algorithm PSSG (Projected Scaled Sub-Gradient) [17], which is designed to solve optimization problems with L1 regularization on parameters. More importantly, PSSG is scalable because it uses the quasi-Newton strategy with a line search that is robust for common functions. Let \( L \) be the log-likelihood of the model. According to the limited-memory BFGS [17] updates for the quasi-Newton method, the gradients of model parameters \( \theta^{time}, \theta^{user} \) and \( \theta^{seq} \) are provided as follows.

\[
\frac{\partial L}{\partial \theta_{t,z}} = d(t, z) - \sum_{j=1}^{[D_z]} a_{u_j,t,z}
\] (12)

\[
\frac{\partial L}{\partial \theta_{u,z}} = d(u, z) - \sum_{j=1}^{[D_u]} a_{u_j,t,z}
\] (13)

\[
\frac{\partial L}{\partial \theta_{v,z}} = d(v, z) - \sum_{j=1}^{[D_v]} a_{u_j,t,z}
\] (14)

where \( d(t, z) \) is the number of activities assigned to topic-region \( z \) on time \( t \); \( d(u, z) \) represents how many activities are assigned to topic-region \( z \) in \( D_u \); \( d(v, z) \) denotes the number of activities whose predecessor set contains the spatial item \( v \) assigned to topic-region \( z \); \( D_v \) denotes the set of activities occurring on time \( t \); \( D_u \) is the set of activities whose predecessor set contains the spatial item \( v \); \( u_j \) denotes the user who generates the \( j^{th} \) activity record and \( t_j \) is the occurring time of the \( j^{th} \) activity in the corresponding set.

Similarly, the gradients of model parameters \( \phi^h \) and \( \phi^{topic} \) are computed as follows:

\[
\frac{\partial L}{\partial \phi_w} = d(w) - \sum_{z=1}^{K} d(z) \times \beta_{z,w}
\] (15)

\[
\frac{\partial L}{\partial \phi_{z,w}} = d(z, w) - d(z) \times \beta_{z,w}
\] (16)

where \( d(w) \) is the number of activities where the word \( w \) appears, and \( d(z, w) \) is the number of activities where the word \( w \) is assigned to the topic-region \( z \).

For geographical modeling, the parameters can be obtained in closed form as Equation 17 and 18:

\[
\mu_z = \frac{1}{d(z)} \sum_{j=1}^{[D_z]} I_j
\] (17)

\[
\Sigma_z = \frac{1}{d(z)-1} \sum_{u=1}^{[U]} \sum_{t=1}^{[T]} (I_{u,t}-\mu_z)(I_{u,t}-\mu_z)^T
\] (18)

where \( D_z \) represents the set of activities assigned to topic-region \( z \) and \( I_j \) is the location where the \( j^{th} \) activity in \( D_z \) occurs.

**Parallel Model Inference.** We infer the TPM model with a parallel algorithm on the distributed GraphLab framework [19] and parallel gradient descent learning framework PSSG (Projected Scaled Sub-Gradient) [17]. PSSG is scalable because it not only uses the quasi-Newton strategy with line search that is robust to common functions, but also adopts the multi-core parallel processing strategy. There are two steps in the Gibbs EM algorithm: Gibbs sampling and gradient descent learning. We decompose the inference procedure of TPM into a two-step parallel processing. In the E step, we implement the Gibbs sampling algorithm in the GraphLab framework [19]. In the M step, PSSG [17]
We adopt the geometric mean for the probability of topic \( z \) where we return the top-

with this two-step parallel processing has been proven being able to achieve satisfying efficiency and scalability on large data in [26].

5 SPATIAL ITEM RECOMMENDATION USING TPM

Once we have estimated the model parameter set \( \Theta \), given a querying user \( u_q \) at time \( t_q \), we first retrieve the spatial item predecessor set \( P_q \) for \( u_q \). Then, we compute the probability of user \( u_q \) choosing each unvisited spatial item \( v \) with description \( W_v \) at location \( I_v \), as in Equation 19. Then, we return the top-\( k \) spatial items with the highest probabilities as recommendations.

\[
P(I_v, W_v | \Theta, u_q, t_q, P_q) = \sum_{z=1}^{K} P(I_v, W_v, z | \Theta, u_q, t_q, P_q) = \sum_{z=1}^{K} \alpha_{uq,tq,z} W_{v}^{\left\lceil \frac{1}{\gamma_{z}} \right\rceil} y_{z,I_v} \tag{19}
\]

In Equation 19, \( W_v \) denotes the content words describing item \( v \) and \( w_{v,n} \) is the \( n^{th} \) word in \( W_v \). We adopt the geometric mean for the probability of topic \( z \) generating the word set \( W_v \), considering that the number of content words is different for different spatial items.

To accelerate the online recommendation process, we propose a ranking framework in Equation 20 which separates the offline computation from the online calculation to the maximum extent.

\[
S(q, v) = \sum_{z=1}^{K} F(z, v) W(q, z) \tag{20}
\]

\[
F(z, v) = \left\lfloor \frac{\left| W_v \right|}{n-1} \beta_{z,w_{v,n}} \right\rfloor \times y_{z,I_v}, \quad W(q, z) = \alpha_{uq,tq,z}
\]

where \( F(z, v) \) represents the offline scoring part which denotes the score of spatial item \( v \) with respect to dimension \( z \). This part is computed offline since it is independent from the query \( q = (u_q, t_q) \). On the other hand, \( W(q, z) \) is inferred in the online part, denoting the preference of query \( q \) on dimension \( z \). Note that the main time-consuming components of \( W(q, z) \) are also computed offline (e.g., \( \theta_{time}, \theta_{user} \) and \( \theta_{seq} \)). This design enables separation the online and offline computations to significantly reduce query time.

When a query \( q = (u_q, t_q) \) arrives, we first compute the query preference weight on each dimension (i.e., \( W(q, z) \)), and then aggregate \( F(z, v) \) over each dimension with the weight \( W(q, z) \) for each spatial item. At last, \( k \) spatial items with the highest scores are returned as the query results. TPM is trained offline, while recommendation performed online is a combination process of the various factors. This scheme guarantees a quick response.

Recommending Cold-start Items. Note that, for cold-start items, our TPM model can still accurately recommend them to the right users according to their location \( I_v \) and textual contents \( W_v \). Different from the score function proposed in [27], the item ID \( v \) is not required in Equation 20. This is why our TPM is able to support cold-start recommendation.

5.1 Efficient Top-k Recommendation

Although the time cost of online recommendation for each query is largely reduced by the offline pre-computation, the online computation is still inefficient when there are a large number of spatial items. This is because the online computation needs to calculate a ranking score for every single candidate item with respect to the query. Thus, we aim to further improve the efficiency of top-\( k \) recommendation by reducing the search space.

One straightforward solution for pruning the item search space is to index spatial items using a tree structure such as R-Tree or Metric Tree, which is widely used in nearest-neighbor search problem in metric spaces. However, according to Equation 20, we transform both queries and items into \( K \)-dimensional vectors \( \vec{q} \) and \( \vec{v} \) (with \( W(q, z) \) and \( F(z, v) \) on each dimension respectively) and measure
the similarities between \( \mathbf{q} \) and \( \mathbf{v} \) using the inner-product, which is different from distance functions in metric spaces (e.g., Euclidean distance and Cosine similarity). For instance, the inner-product lack the basic property of coincidence. In particular, the Euclidean distance of a point to itself is 0 while the inner-product of a point \( \mathbf{v} \) to itself is \( ||\mathbf{v}||^2 \), which may be high or low depending on the length of \( \mathbf{v} \). Therefore, it is infeasible to directly apply the techniques for nearest-neighbor search in our problem.

Ram et al. proposed a technique for maximum inner-product search in [20] based on an adapted metric tree and Yin et al. proposed a solution by extending the Threshold-based Algorithm (TA) in [33, 35]. However, these two solutions suffer from the curse of the dimension \( K \). According to the analysis in [18], the efficiency of the techniques based on a tree index structure is \( O(K^{12}) \). The TA algorithm needs to maintain and access \( K \) sorted lists of items and frequently update the threshold, which makes it slow when \( K \) is large.

Locality Sensitive Hashing (LSH) [7] based techniques are common and successful in industrial practice for solving the KNN problem efficiently. Both the running time and the accuracy guarantee of LSH based KNN are in a way independent of the dimensionality of the data. Furthermore, LSH is massively parallelizable, which makes it ideal for large modern datasets. Although LSH is popular in both Euclidean distance and Cosine similarity, there are few work extending LSH to the Maximum Inner-Product Search (MIPS). LSH involves constructing hashing functions \( h \) such that the probability of \( h(\mathbf{q}) = h(\mathbf{v}) \) is equivalent to the similarity between the query \( \mathbf{q} \) and the item \( \mathbf{v} \), denoted as \( S(\mathbf{q}, \mathbf{v}) \). For any similarity function to admit a locality sensitive hash function family, the distance function (e.g., \( D(\mathbf{q}, \mathbf{v}) = 1 - S(\mathbf{q}, \mathbf{v}) \)) must satisfy the triangle inequality [20]. However, if similarity is measured using inner-product, the distance measure does not satisfy the condition. That is, LSH cannot be directly applied to MIPS.

Inspired by [21], we apply two different hash functions to the spatial items and the queries respectively (i.e., \( h_1 \) for each spatial item \( \mathbf{v} \) and \( h_2 \) for each query \( \mathbf{q} \)), which is called asymmetric LSH (ALSH). The main idea of ALSH is to transform the MIPS into classic nearest neighbor search by introducing two hashing functions so that the probability of new collision event \( h_2(\mathbf{q}) = h_1(\mathbf{v}) \) satisfies the conditions in the definition of KNN for \( S(\mathbf{q}, \mathbf{v}) = \mathbf{q}^T \mathbf{v} \).

We present the ALSH algorithm in Algorithm 1, where we apply two hash functions \( h_1(\mathbf{v}) \) and \( h_2(\mathbf{q}) \) to spatial items and queries respectively. In particular, \( h_1(\mathbf{v}) \) appends \( m \) scalers of the form \( ||\mathbf{v}||^2 \) at the end of the vector \( \mathbf{v} \), while \( h_2(\mathbf{q}) \) simply appends \( m \) “0.5” to the end of the vector \( \mathbf{q} \).

Algorithm 1: The Algorithm of ALSH

**Input:** all the spatial items \( V \) and a given query \( \mathbf{q} \) (both \( \mathbf{q} \) and each item \( \mathbf{v} \) are represented by a vector over \( K \) dimensions);

**Output:** \( k \) spatial items with largest \( S \) in Equation 20;

1. **Preprocessing:**
   1. Scale each \( \mathbf{v} \in V \) to have \( ||\mathbf{v}||_2 \leq 1 \);
   2. Append \( m \) scalars to each \( \mathbf{v} \) as: \( h_1(\mathbf{v}) = [\mathbf{v}; ||\mathbf{v}||_2^2; ||\mathbf{v}||_2^4; \ldots; ||\mathbf{v}||_2^{2m}] \);
   3. Use hash function 22 to create hash tables for \( V \);

2. **Querying:**
   1. Append \( m \) 0.5 to the query \( \mathbf{q} \): \( h_2(\mathbf{q}) = [\mathbf{q}; 0.5; 0.5; \ldots; 0.5] \);
   2. Apply hash function 22 on the transformed query to probe buckets to find top-\( k \) items;

3. **Return** the found top-\( k \) items;
According to [21], we have
\[
\arg\max_{\bar{v} \in \{\bar{v}\}} q^T \bar{v} = \arg\min_{v \in V} ||h_2(q) - h_1(\bar{v})||_2
\]  
(21)

This operation connects MIPS with approximate near neighbor search. Therefore, the LSH can then be applied to solve the problem. For a vector \(\bar{x}\), the hash function proposed in [7] is applied in TPM, as follows.
\[
h_{a,b}(x) = \left\lfloor \frac{a^T \bar{x} + b}{r} \right\rfloor
\]  
(22)

where \(r\) is a fixed real number. There are three parameters in Algorithm 1: \(I\), \(m\) and \(r\). According to empirical analysis in [21], we set \(I = 0.83\), \(m = 3\) and \(r = 2.5\). \(a\) is a random vector with each component generated from i.i.d. normal, i.e., \(a_i \sim N(0,1)\), and \(b\) is a scalar generated uniformly at random from \([0,r]\).

Inspired by the bound analysis of L2LSH in [7], we can conclude that with this algorithm we can construct data structures with worst case \(O(n^{\rho} \log n)\) query time guarantees. According to the analysis in [21], there always exist \(\rho < 1\). This way, we obtain a sublinear query time algorithm.

6 EXPERIMENT

In this section, we first present the experiment settings and then demonstrate the experimental results which include the recommendation effectiveness, impact of factors and recommendation efficiency.

6.1 Experimental Settings

6.1.1 Datasets. We conducted our experiments on two real datasets (Foursquare and Twitter) and one large synthetic dataset. The basic statistics of them are shown in Table 3. The two real datasets are publicly available\(^1\).

Foursquare. This dataset contains the check-in history of 4,163 users who live in California, USA, between Dec 2009 and Jul 2013. Each check-in activity contains the user-ID, item-ID, item-location, item-content and a check-in time.

Twitter. This dataset is based on the publicly available Twitter dataset in [5] which does not contain the category and tag information about spatial items. Twitter supports third-party location sharing services like Foursquare and Gowalla (where users of these services choose to share their check-ins on Twitter). Thus, we crawled the content information from Foursquare with the publicly available API\(^2\). Each record has the same format as the Foursquare dataset.

Synthetic Dataset. To evaluate the online recommendation efficiency of ALSH when the number of spatial items is large, a large synthetic dataset was created following the distribution characteristics of the Foursquare dataset. There are 4,163 users, 121,142 spatial items and 483,813 check-ins on the Foursquare dataset. Approximately, every user has 116 check-ins and every spatial item is associated with 4 check-ins on average. To mimic the data sparsity of the real data, we generate a synthetic dataset with 200 million check-ins, 50 million spatial items and 1.72 million users.

6.1.2 Comparative Approaches. We aim to evaluate both the recommendation effectiveness and the efficiency of generating online recommendations.

To evaluate the effectiveness, we compared TPM with the following four methods which are the state-of-the-art spatial item recommendation techniques. Note that, the temporal cyclic patterns are integrated inside the model for TPM in the evaluation part. To evaluate the improvement brought by integrating the cyclic patterns inside the model, we compare TPM with TPM\(^1\). TPM\(^1\) represents the method integrating temporal cyclic patterns with the time indexing scheme outside the model.

\(^1\)https://sites.google.com/site/dbhongzhi/
\(^2\)http://developer.foursquare.com
Table 3. Statistics of The Three Datasets

<table>
<thead>
<tr>
<th></th>
<th>Foursquare</th>
<th>Twitter</th>
<th>Synthetic</th>
</tr>
</thead>
<tbody>
<tr>
<td># of users</td>
<td>4,163</td>
<td>114,508</td>
<td>1.72 million</td>
</tr>
<tr>
<td># of items</td>
<td>121,142</td>
<td>62,462</td>
<td>50 million</td>
</tr>
<tr>
<td># of check-ins</td>
<td>483,813</td>
<td>1,434,668</td>
<td>200 million</td>
</tr>
</tbody>
</table>

GE: GE [28] is a generic graph-based embedding model. It first jointly embeds four relational graphs (Item-Item representing sequential effect, Item-Region denoting geographical influence, Item-Time describing temporal cyclic effect and Item-Word defining semantic influence) into a shared low dimensional space and then makes recommendation by combining the four factors through a equal-weight linear model.

SPORE: SPORE [27] makes recommendation based on sequential influence and personal interests. It also introduces a latent variable topic-region to combat the data sparsity problem. However, it obtains the topic-regions by mining the textual contents and co-occurrence information of items while TPM learns the topic-regions with textual contents and location information of items.

Category based hidden Markov model (HMM): HMM [4] uses a mixed hidden Markov model to predict the category of the user’s activity at the next step and then predicts the most likely location by incorporating the estimated category distribution and spatial-temporal influence through estimated parameters for each factor. By modeling the category level, HMM significantly reduces the modeling and prediction space compared with the classic Markov model.

LORE: LORE [42] first predicts the probability of a user visiting a location by Additive Markov Chain (AMC) which exploits the sequential effect by adding the influence of the user’s recently visited locations. LORE then fuses sequential influence with geographical influence and social influence by multiplying them together. Note that, the social influence is not explored on the Twitter dataset as there is no network information available on the dataset.

LRT: Gao et al. focus on exploring the affect of temporal cyclical patterns in location recommendation [10]. The original user-location matrix is divided into sub-matrices according to the predefined temporal states. Then, each sub-matrix is factorized into the user check-in preference and the location characteristics.

GT-BNMF: GT-BNMF [15, 16] is a Geographical-Topical Bayesian non-negative Matrix Factorization framework, which integrates all the information except the temporal information, such as geographical influence, personal interest, popularity effect and content effect in a joint manner.

To further validate the benefits brought by considering users’ personal interests, exploiting the sequential/cyclic influence and considering the influence of recently visited items rather than the latest one only, we designed four variant versions of our model. TPM-V1 is the first variation of TPM in which we do not consider users’ personal interests; TPM-V2 is the second simplified version where the sequential influence is not exploited; TPM-V3 does not consider the temporal cyclic effect in making recommendations; and the last variant, TPM-V4 is the simplified version which unifies personal interests with only the sequential influence of the latest visited items.

To evaluate the online recommendation efficiency of ALSH in TPM, we compared it with two baseline methods. The first method is the linear-scan (LS) algorithm without any accelerating scheme. The second one is the threshold algorithm (TA) [33]. This algorithm is able to find the exact top-k results without scanning all spatial items.
6.1.3 Evaluation Methods. Given a user profile, namely a collection of user activities, we first extracted the activity sequence of each user. Then, we used the first 80% of activities in the sequence for each user as the training dataset $D_{train}$ and the remaining activities as the test dataset $D_{test}$. For the data in $D_{train}$, we abstracted the last 20% of activities in the sequence for each user as the validation set. This validation set is used for parameter tuning. To evaluate the recommendation methods, we adopted the evaluation methodology and measurement $Hits@k$ widely used in [12, 25, 30, 35, 40]. Specifically, for each test case $(u, v, W_v, t, P)$ in $D_{test}$ as well as its corresponding query $q$:

1. We computed the ranking score for item $v$ and all other unvisited spatial items.
2. We formed a ranked list by ordering all of these items according to their ranking scores. Let $p$ denote the position of $v$ within this list. The best result is that $v$ precedes all the other unvisited spatial items which means that $p = 1$.
3. We formed a top-$k$ recommendation list by picking the $k$ top ranked items from the list. If $p \leq k$ (i.e., the ground truth item $v$ appears in the top-$k$ recommendation results), we have a hit. Otherwise, we have a miss.

$Hits@k$ is computed as shown in Equation 23. We define $hit@k$ for a single test case as either the value 1, if we have a hit, or the value 0 if we have a miss. The overall $Hits@k$ is defined by averaging over all test cases. $\#hit@k$ denotes the number of hits in the test set and $|D_{test}|$ is the number of all test cases.

$$Hits@k = \frac{\#hit@k}{|D_{test}|}$$ (23)

Note that, both the Foursquare and Twitter datasets have a low density (i.e., the densities of user-item matrix are 0.55% and 0.02% for Foursquare and Twitter datasets, respectively), which usually results in low accuracy values [39]. In addition, the spatial items in the test data of each user may represent only a small portion of POIs that the user is truly interested in. Thus, the low accuracy values are common and reasonable. In this paper, we focus on the relative improvements we achieved, instead of the absolute values.

6.2 Recommendation Effectiveness

6.2.1 Performance with Well-Tuned Parameters. In this section, we first present the experimental results of all the recommendation methods with well-tuned parameters on the validation set. There are two parameters in TPM, namely, the time period threshold ($\Delta T$) and the number of topic-regions ($K$). We tuned these parameters on the validation set. We first present the experimental results with optimal parameter settings: (1) the optimal time period thresholds are 0.5 day for Twitter dataset and 0.2 day for Foursquare dataset; (2) the optimal values of $K$ are 110 for the Twitter dataset.

Fig. 4. Performance on Foursquare and Twitter Datasets
Wang, W. et al

and 120 for the Foursquare dataset. Figures 4(a) and 4(b) report the recommendation effectiveness on the Foursquare and Twitter datasets, respectively. From the results, we observe that the accuracy values gradually increase with the increasing value of $k$. This is because, by returning more spatial items, it is more likely to discover the ones that users would like to visit. Note that we show only the performance when $k$ is set between 2 and 20. Greater values of $k$ are usually ignored for the top-$k$ recommendation task.

TPM performs better than TPM$^1$ on both datasets. This is due to that TPM is able to combat the data sparsity problem of the three-slice time indexing scheme in TPM$^1$. As the check-in data on two datasets are very sparse, the recommendation accuracies for all the comparing methods are low. However, TPM makes a significant improvement compared with the other competitor methods. On the Twitter dataset, the improvements, in terms of Hits@10, are 13.40%, 51.87%, 67.48%, 84.64% and 116.81% compared with GE, LRT, LORE, HMM and GT-BNMF, respectively, which clearly demonstrate the advantages of our proposed TPM model with respect to other competitor models.

Several observations are made from the results in Figures 4(a) and 4(b): 1) On the Foursquare dataset, all the methods perform better than on the Twitter dataset due to that the user-item matrix on Twitter dataset is much sparser than the one on the Foursquare dataset. 2) TPM outperforms GE on both datasets. The possible reason is that GE overlooks personalization and weight in integrating different factors. On one hand, GE captures sequential patterns through a shared Item-Item graph embedding for all users. In this way, it assumes equivalent transition probabilities between items for all users, and ignores personalization. On the other hand, GE combines the sequential effect, cyclic patterns and personalization through a simply linear combination without any weight scheme. By contrast, TPM in our paper integrates these three factors in a unified and personalized way by learning a personalized weight for each factor. 3) TPM outperforms SPORE significantly on both datasets. This may due to two reasons. The first reason is that TPM integrates the cyclic patterns and the second reason is that TPM obtains topic-region in a more proper way. 4) Both LORE and HMM drop behind TPM, showing the advantage of seamlessly integrating the multiple factors into a unified framework by avoiding the inference of mixture weights for each factor, especially when the activity data of each user is sparse. LORE considers the sequential effect and other factors by simply multiplying them together, which is oversimplified. HMM accomplishes the fusion by inferring a weight for each factor which is inaccurate when the data is sparse. By contrast, our TPM adds the effect of all the factors in the exponential space to avoid the inference of weight for each factor to gain improved robustness and accuracy. 5) TPM outperforms GT-BNMF on both datasets, demonstrating the benefits brought by considering sequential influence in personalized spatial item recommendation.

6.2.2 Parameter Tuning. To tune the two parameters in TPM, i.e. $K$ and $\Delta T$, we tested different setups for them on the validation set. Due to space constraints, we have only shown the experimental results for the top-10 recommendation using the Twitter dataset. We tested the performance of TPM by varying the number of topic-regions $K$ from 90 to 140 and the time threshold $\Delta T$ from 0.2 days to 0.7 days. The results are presented in Table 4. From the results, we observe that the performance first improves quickly with the increase of the number of topic-regions and then the increment becomes small. The number of the topic-regions represents the model complexity. Thus, when $K$ is too small, the model has limited ability to describe the data. However, when $K$ exceeds a threshold (e.g., $K = 100$), the model is complex enough to handle the data. At this point, it is less helpful to improve the model performance by increasing $K$. A similar trend is observed for the time period threshold. There are more predecessors when $\Delta T$ becomes larger according to Definition 3.3. Thus, the performance improves with the increase of $\Delta T$. However, when $\Delta T$ is larger than a threshold (e.g., $\Delta T = 0.5$ day), the added predecessors have little influence on the recommendation performance. As
a result, we chose $K = 120$ and $\Delta T = 0.5$ day as the best trade-off between accuracy and efficiency on the Twitter dataset.

<table>
<thead>
<tr>
<th>$\Delta T$</th>
<th>90</th>
<th>100</th>
<th>110</th>
<th>120</th>
<th>130</th>
<th>140</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>0.0786</td>
<td>0.0807</td>
<td>0.0820</td>
<td>0.0828</td>
<td>0.0831</td>
<td>0.0832</td>
</tr>
<tr>
<td>0.3</td>
<td>0.0802</td>
<td>0.0821</td>
<td>0.0836</td>
<td>0.0844</td>
<td>0.0848</td>
<td>0.0848</td>
</tr>
<tr>
<td>0.4</td>
<td>0.0814</td>
<td>0.0832</td>
<td>0.0847</td>
<td>0.0856</td>
<td>0.0857</td>
<td>0.0859</td>
</tr>
<tr>
<td>0.5</td>
<td>0.0824</td>
<td>0.0842</td>
<td>0.0855</td>
<td><strong>0.0864</strong></td>
<td>0.0867</td>
<td>0.0866</td>
</tr>
<tr>
<td>0.6</td>
<td>0.0825</td>
<td>0.0843</td>
<td>0.0855</td>
<td>0.0865</td>
<td>0.0867</td>
<td>0.0867</td>
</tr>
<tr>
<td>0.7</td>
<td>0.0825</td>
<td>0.0844</td>
<td>0.0857</td>
<td>0.0865</td>
<td>0.0867</td>
<td>0.0868</td>
</tr>
</tbody>
</table>

Table 4. Impact of Parameters

### 6.3 Impact of Different Factors

In this subsection, to validate the benefits brought by exploiting the users’ personal interests, exploiting the sequential/cyclic influence and considering the influence of recently visited items rather than the last one only, we compared our TPM model with its four variant versions: TPM-V1, TPM-V2, TPM-V3 and TPM-V4 respectively. We also studied the impact of the number of topic-regions $K$ and the time threshold $\Delta T$.

The results of comparing TPM with the four variant versions on both datasets are shown in Figure 5. The results show that TPM consistently outperforms the four variant versions on both datasets, indicating the benefits brought by each factor, respectively. For instance, the performance gap between TPM and TPM-V2 validates the effectiveness of leveraging the sequential influence of visited items into recommendation. The improvement of TPM over TPM-V3 on both datasets shows the advantage of exploiting the temporal cyclic influence. Another observation is that TPM-V2, TPM-V3 and TPM-V4 outperform TPM-V1 significantly, showing that the users’ personal interests play the most important role in spatial item recommendation on both LBSN data sets. This is because of that most LBSN data has a low sampling rate in terms of time. Like we analyzed before, only 10% of users have more than 58 check-in records over a 12-month period. Given such sparse data in terms of time, it’s hard to mine useful temporal patterns. That’s why personal interests bring the more significant improvement compared with the other two time related components. What is also worth noting is that the performance gap between TPM and TPM-V2 on the Twitter dataset is larger.
than that on the Foursquare dataset. This can be explained by the fact that the sequential information on the Twitter dataset is much denser than that on the Foursquare dataset.

### 6.4 Recommendation Efficiency

This experiment is to evaluate the efficiency of our proposed online recommendation algorithm ALSH on both the real-life and large-scale synthetic datasets. We compared ALSH with two algorithms. The first algorithm is the threshold algorithm (TA) [33] developed for online recommendation. It pre-computes a sorted list for each dimension $z$ in which items are sorted according to their scores on $z$ (i.e., $F(z,v)$), and also maintains a priority queue of the $K$ sorted lists that controls which sorted list to access in the next. The algorithm has the nice property of terminating early without scanning all spatial items. Specifically, it terminates when the ranking score of the $k$-th item in the result list is higher than the threshold score. The other algorithm is the linear-scanning method (LS) that linearly scans all spatial items by computing a ranking score for each item according to Equation 20 and selects top-$k$ ones with highest ranking scores. All the online recommendation algorithms were implemented in Java (JDK 1.7) and ran on a Windows Server 2012 with 256G RAM.

Table 5 presents the average online efficiency of the three different methods on the Foursquare dataset. On average, our proposed ALSH produces top-10 recommendations in 2.73ms. From this results, we observe that 1) ALSH outperforms LS and TA significantly, which demonstrates that ALSH-based query processing technique is much more efficient; 2) the time costs of all methods increase with the increasing number of recommendations ($k$).

<table>
<thead>
<tr>
<th>Methods</th>
<th>Online Recommendation Time Cost (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$k = 1$</td>
</tr>
<tr>
<td>ALSH</td>
<td>2.27</td>
</tr>
<tr>
<td>TA</td>
<td>4.62</td>
</tr>
<tr>
<td>LS</td>
<td>20.15</td>
</tr>
</tbody>
</table>

Table 5. Recommendation Efficiency on Foursquare Dataset

To further evaluate the scalability of TPM, another experiment was conducted on the synthetic dataset. We varied the number of candidate spatial items from 10 million to 50 million based on the fact that, given a query and the number of recommendations ($k$), the efficiency of producing online recommendation mainly depends on the number of the available spatial items. Figure 6 presents the time costs for producing top-10 recommendations by varying number of available items from 10
millions to 50 millions. From Figure 6, we can see that ALSH reduces the processing time for each online query significantly compared with both LS (1.34s vs 11.90s) and TA (1.28s vs 4.15s) when the number of spatial items is 50 million. This improvement is crucial to enhancing the online users’ experience in the real-life scenario where the number of spatial items is very large.

**Discussion about the Accuracy of ALSH:** LS returns the exact top-$k$ spatial items with highest ranking scores. However, ALSH finds the approximate top-$k$ spatial items with highest ranking scores. To evaluate the accuracy of ALSH in making recommendations, we compared the Hits@$k$ values of ALSH with the ones of LS. Figure 7 presents comparison results on the Foursquare dataset. From Table 5 and Figure 7, we can see that ALSH reduces the online query time significantly (about 87.80%) at the cost of a minor accuracy lose (8.00%) in producing top-10 recommendations, compared with LS.

### 6.5 Test for Cold Start Items

As we argued before, different from our preliminary work in [27], TPM mines the topic-regions from location information instead of the co-occurrence of items. In this way, TPM is able to model the cold-start items without visiting information as long as their location information is available. In this section, we conduct experiments to study the effectiveness of different recommendation algorithms handling the cold start items. Cold start items in the domain of recommendation refer to the items which have not been visited by any users, or have been visited by only a few users. We test the recommendation effectiveness for cold-start items on both Foursquare and Twitter dataset and present the results in Figure 8 and Figure 9 respectively. Figure 8(a) and Figure 9(a) report the result for...
items with less than 5 activity records, and Figure 8(b) and Figure 9(b) report the result for items without any activity record.

We observe that, although the recommendation effectiveness of all algorithms decrease to various degree in recommendation for cold start items, our proposed TPM still outperforms the others on both datasets. Note that it is very hard to capture users’ preference on the cold-start items from extremely limited number of visiting records. However, TPM is able to find these preference by mining users’ preference over the content information and location information associated with the cold-start items. TPM-V3 and SPORE both exploit the sequential influence and personal interests with topic-regions. However, TPM-V3 outperforms SPORE in recommending cold-start items. This is due to that TPM-V3 changes the way of mining the topic-regions compared with SPORE. It mines the topic-regions from location information instead of the co-occurrence of items as cold-start items have few co-occurrence with other items. LRT performs the worst on both datasets as it is a memory-based method which suffers from severe data sparsity, while the other methods either explore the latent classes or integrate the geographical or content information which can relieve the data sparsity problem to a great extent.

7 CONCLUSION

In this paper, we proposed a novel temporal personalized recommendation framework (TPM). TPM introduces a novel latent variable topic-region to model and fuse the sequential influence, cyclic patterns and personal interests in the latent space. A topic-region corresponds to both a semantic topic (i.e., a soft cluster of words) and a geographical region (i.e., a soft cluster of locations). The advantages of modeling sequential effect at the topic-region level include a significantly reduced prediction space, an effective alleviation of data sparsity and a direct expression of the semantic meaning of users’ spatial activities. To explore the various cyclic patterns in a unified way, TPM introduces two methods. The first method encodes the various cyclic patterns into a unified binary code outside the model. This method suffers from the data sparsity in each time slice. To deal with the data sparsity problem, the second method slices the time according to each cyclic pattern separately and explores these patterns in a joint additive model. To seamlessly fuse sequential effect and personal interests in a unified and principled way, we adopt the sparse additive modeling technique to add them to exponential space thus avoiding the inference of mixture weights for each factor. Further, we designed an asymmetric Locality Sensitive Hashing (ALSH) technique to speed up online top-k recommendations by extending the traditional LSH. Extensive experiments were conducted to evaluate the performance of TPM on two real datasets and one large-scale synthetic dataset. The performance of TPM in recommending cold-start items is also evaluated. The results demonstrate the advantages of TPM in terms of both recommendation effectiveness and efficiency.

ACKNOWLEDGMENTS

This work was supported by ARC Discovery Early Career Researcher Award (Grant No. DE160100308), ARC Discovery Project (Grant No. DP170103954) and New Staff Research Grant of The University of Queensland (Grant No. 613134).

REFERENCES


[38] Zhijun Yin, Liangliang Gao, Jiawei Han, Jiebo Luo, and Thomas S. Huang. 2011. Diversified Trajectory Pattern Ranking in Geo-tagged Social Media. In *SIAM*. 980–991.


