Joint Modeling of User Check-in Behaviors for Real-time Point-of-Interest Recommendation

HONGZHI YIN, The University of Queensland
BIN CUI, Peking University
XIAOFANG ZHOU, WEIQING WANG, ZI HUANG, and SHAZIA SADIQ,
The University of Queensland

Point-of-Interest (POI) recommendation has become an important means to help people discover attractive and interesting places, especially when users travel out of town. However, the extreme sparsity of a user-POI matrix creates a severe challenge. To cope with this challenge, we propose a unified probabilistic generative model, the *Topic-Region Model (TRM)*, to simultaneously discover the semantic, temporal, and spatial patterns of users' check-in activities, and to model their joint effect on users' decision making for selection of POIs to visit. To demonstrate the applicability and flexibility of TRM, we investigate how it supports two recommendation scenarios in a unified way, that is, hometown recommendation and out-of-town recommendation. TRM effectively overcomes data sparsity by the complementarity and mutual enhancement of the diverse information associated with users' check-in activities (e.g., check-in content, time, and location) in the processes of discovering heterogeneous patterns and producing recommendations. To support real-time POI recommendations, we further extend the TRM model to an online learning model, TRM-Online, to track changing user interests and speed up the model training. In addition, based on the learned model, we propose a clustering-based branch and bound algorithm (CBB) to prune the POI search space and facilitate fast retrieval of the top-*k* recommendations.

We conduct extensive experiments to evaluate the performance of our proposals on two real-world datasets, including recommendation effectiveness, overcoming the cold-start problem, recommendation efficiency, and model-training efficiency. The experimental results demonstrate the superiority of our TRM models, especially TRM-Online, compared with state-of-the-art competitive methods, by making more effective and efficient mobile recommendations. In addition, we study the importance of each type of pattern in the two recommendation scenarios, respectively, and find that exploiting temporal patterns is most important for the hometown recommendation scenario, while the semantic patterns play a dominant role in improving the recommendation effectiveness for out-of-town users.

Categories and Subject Descriptors: H.3.3 [**Information Search and Retrieval**]: Information filtering; H.2.8 [**Database Applications**]: Data mining; J.4 [**Computer Applications**]: Social and Behavior Sciences

General Terms: Algorithms, Design, Experimentation, Performance

This work is partially supported by ARC Discovery Early Career Researcher Award (Grant No. DE160100308) and ARC Discovery Project (Grant No. DP140103171). It is also partially supported by the National Natural Science Foundation of China (Grant Nos. 61572335, 61572039, 61232006, 61272155), 973 program under No. 2014CB340405, and Shenzhen Government Research Project JCYJ20151014093505032.
Authors' addresses: H. Yin, The University of Queensland, School of Information Technology and Electrical Engineering, QLD 4072, Australia; email: db.hongzhi@gmail.com; B. Cui, Key Lab of High Confidence Software Technologies (MOE), School of EECS, Peking University, Science Building 1, Beijing 100871, China; email: bin.cui@pku.edu.cn; X. Zhou, W. Wang, Z. Huang, and S. Sadiq, The University of Queensland, School of Information Technology and Electrical Engineering, QLD 4072, Australia; emails: {zxf, huang, shazia}@itee.uq.edu.au, weiqingwang@uq.edu.au.

DOI: http://dx.doi.org/10.1145/2873055

Additional Key Words and Phrases: POI, real-time recommendation, location-based service, online learning, efficient retrieval algorithm

ACM Reference Format:
Hongzhi Yin, Bin Cui, Xiaofang Zhou, Weiqing Wang, Zi Huang, and Shazia Sadiq. 2016. Joint modeling of user check-in behaviors for real-time point-of-interest recommendation. ACM Trans. Inf. Syst. 35, 2, Article 11 (October 2016), 44 pages.
DOI: http://dx.doi.org/10.1145/2873055

1. INTRODUCTION

With the rapid development of Web 2.0, and location acquisition and wireless communication technologies, a number of *location-based social networks (LBSNs)* have emerged and flourished, such as Foursquare, Loopt, and Facebook Places, where users can check in at points of interest (POIs) and share life experiences in the physical world via mobile devices. On the one hand, the new dimension of location implies extensive knowledge about an individual's behaviors and interests by bridging the gap between online social networks and the physical world, which enables us to better understand user preferences and design optimal recommender systems. On the other hand, it is valuable to develop the *location recommendation service* as an essential function of LBSNs to encourage users to explore new locations [Ference et al. 2013]. Therefore, developing recommender systems for LBSNs to provide users with POIs has recently attracted increasing research attention. This application becomes more important and useful when users travel to an unfamiliar area, where they have little knowledge about the neighborhood. In this scenario, the recommender system is proposed as *recommendation for out-of-town users* in Ference et al. [2013]. In this article, we aim to offer accurate recommendations for both hometown and out-of-town users by mining their historical activity data in LBSNs.

One of the most important problems for POI recommendation is how to deal with a severe challenge stemming from the extreme sparsity of the user-POI interaction matrix. There are millions of POIs in LBSNs, but a user can only visit a limited number of them. Moreover, the observation of travel locality exacerbates this problem. The observation of travel locality [Levandoski et al. 2012] made on LBSNs shows that most of the users' check-ins are left in their living regions (e.g., home cities). An investigation shows that the check-in records generated by users in their nonhome cities are very few and take up only 0.47% of the check-in records left in their home cities [Scellato et al. 2011]. This observation of travel locality is quite common in the real world [Scellato et al. 2011], aggravating the data sparsity problem with POI recommendation for out-of-town users (e.g., if we want to recommend POIs located in Los Angeles to people from New York City) [Ference et al. 2013; Yin et al. 2014].

The most popular approach in recommender systems is *collaborative filtering* [Adomavicius and Tuzhilin 2005]. There exists a considerable body of research [Levandoski et al. 2012; Ye et al. 2010; Lian et al. 2014; Ference et al. 2013; Gao et al. 2013] that deposited people's check-in history into a user-POI matrix in which each row corresponds to a user's POI-visiting history and each column denotes a POI. A collaborative filtering-based method is then employed by Levandoski et al. [2012], Ye et al. [2010], and Ference et al. [2013] to infer the user's preference regarding each unvisited POI. Based on the core idea of collaborative filtering, similar users of the target user (i.e., those who exhibit similar POI visiting behaviors) are chosen to provide clues for making recommendations. Due to travel locality, most of these similar users are more likely to live in the same region with the target user than other regions. As a recommendation is made by considering POIs visited by similar users, most of the recommended POIs would be located in the target user's hometown. Thus, these CF-based methods cannot be directly applied to the POI recommendations

Table I. A POI and Its Associated Information

Name: Lone Pine Koala Sanctuary	
Location: Longitude: 152.968, Latitude: -27.533	
Categories: Zoo, Park	
Tags: koala, platypus, kangaroo, zoos, wildlife, animal, benches, show	

for out-of-town users [Ference et al. 2013; Yin et al. 2014]. In addition, some recent literature [Lian et al. 2014; Gao et al. 2013] adopted latent factor models (e.g., matrix factorization) to a user-POI matrix to alleviate data sparsity. However, they do not perform well for the out-of-town scenario because most of the check-ins in the training set are hometown check-ins, and there are few common users between a hometown POI and an out-of-town POI, resulting in a very weak correlation between them. Thus, the rating score of a target user to an out-of-town POI predicted by the trained matrix factorization model is not reliable. For example, the predicted rating score of a target user to an out-of-town POI is very low, although the user potentially prefers that POI.

1.1. Joint Modeling of User Check-in Behaviors

To deal with the issue of data sparsity, especially for the out-of-town recommendation scenario, we proposed a unified probabilistic generative model in our previous work [Yin et al. 2015c], namely, the Topic-Region Model (TRM), to simultaneously discover the semantic, temporal, and spatial patterns of users' check-in activities, and model their joint effect on users' check-in behaviors. (1) Semantic Patterns. A recent analysis of the Whrrl dataset shows that the check-in activities of users exhibit a strong semantic regularity [Ye et al. 2011]. In the analysis, Ye et al. studied the diversity of POIs that individual users visit by computing the entropy of semantic categories in their check-ins. The results show that most users have very small entropies. (2) Temporal Cyclic Patterns. As observed in Yuan et al. [2013] and Gao et al. [2013], users' activity contents exhibit strong temporal cyclic patterns in terms of hour of the day or day of the week. For example, a user is more likely to go to a restaurant rather than a bar at lunch time, and is more likely to go to a bar rather than an office at midnight. (3) Spatial Patterns. Many recent studies show that people tend to explore POIs near the ones that they have visited before [Ye et al. 2011]. Thus, POIs visited by users often form spatial clusters, that is, people tend to check in around several centers (e.g., "home" and "office") [Cheng et al. 2012; Lian et al. 2014]. Note that while there are some recent studies [Yin et al. 2014; Yuan et al. 2013; Gao et al. 2013; Lian et al. 2014; Cheng et al. 2012] that exploited one of these patterns to improve POI recommendation, they lack an integrated analysis of their joint effect to deal with the issue of data sparsity, especially in the out-of-town recommendation scenario, due to the difficulty of modeling heterogenous information and discovering multiple types of patterns in a unified way.

As shown in Table I, a POI from Foursquare[1] has both semantic and geographical attributes. Thus, two corresponding components are designed in TRM: User Interest Component (UIC) and User Mobility Component (UMC). We adopt two basic latent variables in these two components: topic and region, which are responsible for generating semantic attributes (e.g., tags and categories) and geographical attributes (e.g., geographical coordinates) of visited POIs, respectively.

The UIC aims to exploit both the contents of POIs and their temporal cyclic effect to model users' interests. An individual's interests (i.e., semantic patterns) are modeled as a distribution over topics. Specifically, we infer an individual's interests according to the semantic contents of the individual's checked-in POIs, respectively. Thus, our model alleviates data sparsity to some extent, especially for the out-of-town recommendation

[1]https://foursquare.com/.

scenario, as the semantic contents play the role of medium that can transfer users' interests inferred at their home regions to out-of-town regions where they are traveling. As the quality of the learned topics is very important for user-interest modeling, we exploit the temporal cyclic patterns of visiting POIs to improve the process of topic discovery. For example, the POIs frequently visited at lunch and dinner are more likely to be restaurants, while the ones visited around midnight are more likely to be nightlife spots.

The UMC is developed to exploit the geographical influence to model users' mobility patterns. In this component, all POIs are divided into several geographical regions according to their locations and check-in frequency. We first compute the probability of each region that an individual user is likely to visit according to the location distribution of the user's historical visited POIs or current location. Then, we infer the probability of each location generated from a region according to the public's check-in behaviors at that region. For example, if a POI v is very popular and frequently checked in at region r, then the probability of region r generating v's geographical coordinate l_v is high. By integrating the region-level popularity of POIs, that is, the wisdom of crowds, the TRM model can alleviate the issue of user-interest drift across geographical regions, to some extent, which indicates that user interests inferred at one region (e.g., home town) cannot always be applied to recommendations at another region. For example, a user u never goes gambling when the user lives in Beijing, China, but when that user travels to Macao or Las Vegas, the individual is most likely to visit casinos.

As a user's decision-making for the selection of POIs to visit is influenced by the joint effect of personal interests and spatial mobility patterns, we propose a joint latent factor *topic-region* to capture this joint effect, which is responsible for generating the IDs of visited POIs. A topic-region represents a geographical area in which POIs have the same or similar semantics.

1.2. Real-Time POI Recommendation

As time goes on, users' interests may change and need different POIs. This requires producing recommendation results in a real-time manner. However, the current TRM developed in Yin et al. [2015c] is incapable of supporting real-time recommendation due to the following reasons. First, although the TRM exploits the temporal cyclic effect of the public visiting POIs to improve topic discovery, it assumes that the individuals' interests are stable and ignores their dynamics in the long term. In reality, they are changing over time, as analyzed in Yin et al. [2015b]. For instance, users will naturally be interested in visiting parenting-related POIs (e.g., the playground and amusement park) after they have a baby, and probably ignore their other interests. For another example, when people move from a city to another city where there are different urban compositions and cultures, their interests will be most likely to change. Accurately capturing this change in a real-time manner has been proven to be commercially very valuable since it indicates visiting and purchasing intents. Second, it is difficult to apply the current TRM for large-scale check-in data that arrives in a stream, since the batch-learning algorithm developed for the TRM in Yin et al. [2015c] needs to run through all check-in data many times (about 1000 iterations), and it is very time-consuming and infeasible. Therefore, to support real-time POI recommendation, in this article, we extend the batch TRM [Yin et al. 2015c] to an online learning model, TRM-Online, which can efficiently process the check-in stream and track changing user interests.

To demonstrate the applicability of TRMs, we investigate how they support two recommendation scenarios in a unified way: (1) hometown recommendation, which assumes that the target user is located in one's home town, that is, to meet users' information needs in their daily life; and (2) out-of-town recommendation, which aims to cater to users when they travel out of town, especially in an unfamiliar region. Real-time recommendation requires that both of the recommendation scenarios should be time-aware [Yuan et al. 2013], location-based [Ference et al. 2013] and personalized, that is, to recommend different ranked lists of POIs for the same target user at different times and locations. Given a querying user u_q with the current location l_q and time t_q (i.e., $q = (u_q, l_q, t_q)$), the naïve approach to produce online top-k recommendations is to first compute a ranking score for each POI, then select k ones with the highest ranking scores. However, when the number of available POIs becomes large, to produce a top-k ranked list using this brute-force method is very time-consuming and slow. To support real-time POI recommendation, we develop efficient retrieval algorithms to speed up the process of online recommendation. Specifically, we propose a clustering-based branch and bound algorithm (CBB) to prune the POI search space and facilitate fast retrieval of top-k recommendations. Compared with the existing fast retrieval algorithms [Koenigstein et al. 2012; Yin et al. 2015b; Teflioudi et al. 2015] such as R-Tree- and Metric-Tree-based query processing techniques, our CBB algorithm has the desirable ability to overcome the curse of dimensionality.

To sum up, this article focuses on the problem of joint modeling user check-in behaviors for real-time POI recommendation. To the best of our knowledge, this is the first work focusing on real-time POI recommendation. This article makes the following contributions:

—We propose a probabilistic generative TRM for joint modeling of users' check-in behaviors by exploiting the semantic, temporal, and spatial patterns in a unified way. To estimate model parameters, we first present a detailed batch inference algorithm (TRM-Batch).
—Based on the TRM-Batch, we propose an online learning model, TRM-Online, to track the dynamics of user interests and speed up model training for supporting real-time recommendation.
—We develop a CBB to prune the POI search space and facilitate fast computation of top-k recommendation.
—To evaluate the performance of our TRMs in real-time POI recommendation, we adopt a new evaluation methodology to simulate a more real recommendation scenario and conduct more extensive experiments, including recommendation effectiveness, recommendation efficiency, and model-training efficiency. We also investigate how TRMs overcome the cold-start problem and study the contribution of each type of pattern to improve recommendation accuracy in the two respective recommendation scenarios.

The remainder of the article is organized as follows. Section 2 details the TRM and its batch-learning algorithm. We present an online learning algorithm for the TRM in Section 3. We present how to deploy TRMs to two typical POI recommendation scenarios and overcome the cold-start problem in Section 4. We propose an efficient retrieval algorithm to support real-time recommendation in Section 5. We describe the experimental setup and report the experimental results in Section 6. Section 7 reviews the related work, and we present our conclusions in Section 8.

2. JOINT MODELING OF USER CHECK-IN ACTIVITIES

In this section, we first formulate the problem definition, then present our proposed TRM.

2.1. Preliminary

For tease of presentation, we define the key data structures and notations used in this article.

Definition 2.1 (POI). A POI is defined as a uniquely identified specific site (e.g., a restaurant or a cinema). In our model, a POI has three attributes: identifier, location, and contents. We use v to represent a POI identifier and l_v to denote its corresponding geographical attribute in terms of longitude and latitude coordinates. It should be noted that many different POIs can share the same geographical coordinate. In addition, there is textual semantic information associated with a POI, such as the category and tag words. We use the notation $\mathcal{W}_v$ to denote the set of words describing POI v, and define $|\mathcal{W}_v| = W_v$.

Definition 2.2 (*User Home Location*). Following the recent work of Li et al. [2012], given a user u, we define the user's home location as the place where the user lives, denoted as l_u.

Note that we assume that a user's home location is "permanent" in our problem. In other words, a home location is a static location instead of a real-time location that is "temporally" related to the user (e.g., the places where the user is traveling). Due to privacy, users' home locations are not always available. For a user whose home location is not explicitly given, we adopt the method developed by Scellato et al. [2011]. This method discretizes the world into $25km$-by-$25km$ cells and finds the one with most of the user's check-ins. Then, the user's home location is defined as the average position of all of the user's check-ins within the cell.

Definition 2.3 (*Check-in Activity*). A check-in activity is represented by a five tuple $(u, v, l_v, \mathcal{W}_v, t)$, which means that user u visits POI v at time t.

As suggested in Yuan et al. [2013] and Gao et al. [2013], human geographical movement exhibits significant temporal cyclic patterns on LBSNs that are highly relevant to the POI contents, and the daily pattern (hours of the day) is one of the most fundamental temporal patterns that reflects a user's visiting behavior. Therefore, we investigate the features embedded in daily patterns in this work, and split a day into multiple equal time slots based on hour. Time and time slot are used interchangeably in this article unless noted otherwise.

Definition 2.4 (*User Document*). For each user u, we create a user document $\mathcal{D}_u$, which is a set of check-in activities associated with user u, and we define $D_u = |\mathcal{D}_u|$. The dataset $\mathcal{D}$ used in our model consists of user documents, that is, $\mathcal{D} = \{\mathcal{D}_u : u \in \mathcal{U}\}$, where $\mathcal{U}$ is the set of users.

Definition 2.5 (*Topic*). Given a collection of words $\mathcal{W}$, a topic z is defined as a multinomial distribution over $\mathcal{W}$, that is, $\boldsymbol{\phi}_z = \{\phi_{z,w} : w \in \mathcal{W}\}$, where each component $\phi_{z,w}$ denotes the probability of topic z generating word w. Generally, a topic is a semantic-coherent soft cluster of words.

Given a dataset $\mathcal{D}$ as the union of a collection of user documents, we aim to provide POI recommendations for both hometown and out-of-town users. We formulate our problem, which takes into account both of the scenarios, in a unified fashion, as follows.

PROBLEM 1 (POI RECOMMENDATION). *Given a user check-in activity dataset $\mathcal{D}$ and a target user u_q with current location l_q and time t_q (i.e., the query is $q = (u_q, t_q, l_q)$), our goal is to recommend top-k new POIs that u_q would be interested in. Given a distance threshold d, the problem becomes an* ***out-of-town recommendation*** *if the distance between the target user's current location and home location (i.e., $|l_q - l_{u_q}|$) is greater than d. Otherwise, the problem is a* ***hometown recommendation****.*

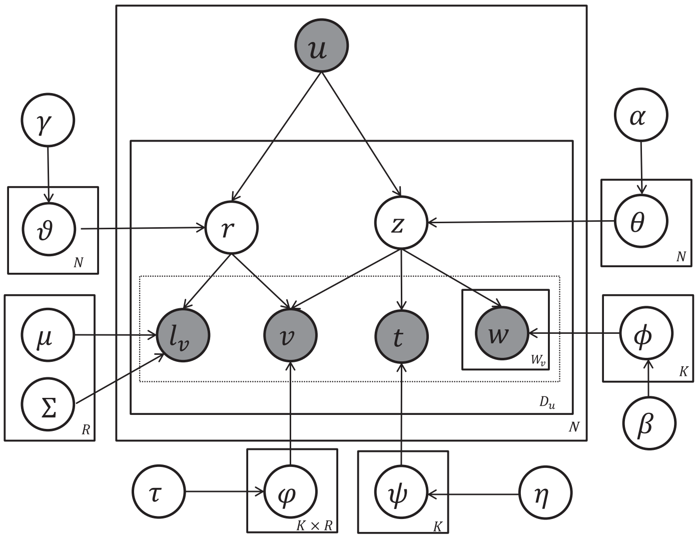


Fig. 1. The graphical representation of the TRM.

Table II. Notations of Parameters Used in TRM

Variable	Interpretation
$\boldsymbol{\vartheta_u}$	the spatial patterns of user u, expressed by a multinomial distribution over a set of regions
$\boldsymbol{\theta_u}$	the interests of user u, expressed by a multinomial distribution over a set of topics
$\boldsymbol{\phi_z}$	a multinomial distribution over words specific to topic z
$\boldsymbol{\psi_z}$	a multinomial distribution over time slots specific to topic z
$\boldsymbol{\varphi_{z,r}}$	a multinomial distribution over POI IDs specific to topic-region (z, r)
$\boldsymbol{\mu_r}$	the mean location of region r
$\sum_{\boldsymbol{r}}$	the location covariance of region r
$\gamma, \alpha, \beta, \eta, \tau$	Dirichlet priors to multinomial distributions
	$\boldsymbol{\vartheta_u}$, $\boldsymbol{\theta_u}$, $\boldsymbol{\phi_z}$, $\boldsymbol{\psi_z}$, and $\boldsymbol{\varphi_{z,r}}$, respectively

Following related studies [Ference et al. 2013; Mok et al. 2010], we set $d = 100km$ in our work, since a distance around $100km$ is the typical radius of human "reach" – it takes about 1h to 2h to drive such a distance.

2.2. Model Structure

To model user check-in activities, we propose a unified probabilistic generative model TRM to simulate the process of the user's decision making for the selection of POIs. Figure 1 shows the graphical representation of TRM; N, K and R denote the number of users, topics, and regions, respectively. We first introduce the notations of our model and list them in Table II. Our input data, that is, users' check-in records, are modeled as observed random variables, shown as shaded circles in Figure 1. As a POI has both semantic and geographical attributes, we introduce two latent random variables, topic z and region r, which are responsible for generating them, respectively. Based on the two latent factors, TRM aims to model and infer users' interests and spatial mobility patterns, as well as their joint effect on users' selection of POIs.

User Interest Modeling. Intuitively, a user chooses a POI by matching one's personal interests with the contents of that POI. Inspired by the early work about user-interest modeling [Liu and Xiong 2013; Yin et al. 2014; Hu and Ester 2013; Hong et al. 2012], the TRM adopts latent topics to characterize users' interests to overcome the data sparsity of the user-word matrix. Specifically, we infer the individual user's interest distribution over a set of topics according to the contents (e.g., tags and categories) of the user's checked-in POIs, denoted as $\boldsymbol{\theta_u}$. Thus, the quality of topics is very important for accurately modeling users' interests. To improve the topic discovery process, we exploit the temporal patterns of visiting POIs or, more exactly, daily patterns. Intuitively, different types of POIs have different temporal patterns of check-ins; two POIs exhibiting similar temporal patterns are more likely to have the same/similar functions and categories than two random ones. For example, the POIs frequently visited at lunch and dinner are more likely to be restaurants, while the ones visited around midnight are more likely to be nightlife spots. Based on this intuition, a topic z in TRM is responsible for simultaneously generating semantic words $\mathcal{W}_v$ and check-in time t. Thus, each topic z in TRM is not only associated with a word distribution $\boldsymbol{\phi_z}$, but also with a distribution over time $\boldsymbol{\psi_z}$. This design enables $\boldsymbol{\phi_z}$ and $\boldsymbol{\psi_z}$ to be mutually influenced and enhanced during the topic discovery process, facilitating the clustering of the words of POIs with similar temporal patterns into the same topic with high probability. To integrate the check-in time information to the topic discovery process, we employ the widely adopted discretization method in Gao et al. [2013] and Yuan et al. [2013] to split a day into hourly-based slots.

In the standard topic models [Blei et al. 2003; Wallach et al. 2009], a document (i.e., a bag of words) contains a mixture of topics, represented by a topic distribution, and each word has a hidden topic label. While this is a reasonable assumption for long documents, for a short document $\mathcal{W}_v$, it is most likely to be about a single topic. We therefore assign a single topic to the document $\mathcal{W}_v$. A similar idea of assigning a single topic to a Twitter post has been used before [Zhao et al. 2011].

User Mobility Modeling. In contrast to users' online behaviors in the virtual world, users' check-in activities in the physical world are limited by travel distance. Thus, it is also important to capture users' spatial patterns (or activity ranges) according to the location distributions of their historical checked-in POIs. The spatial clustering phenomenon indicates that users are most likely to check in a number of POIs that are usually limited to some specific geographical regions [Ye et al. 2011] (e.g., "home" and "office" regions). In this component, all POIs are divided into R regions according to their geographical locations and check-in densities. Following the literature [Liu et al. 2013; Lichman and Smyth 2014], we assume a Gaussian distribution for each region r, and the location for POI v is characterized by $l_v \sim \mathcal{N}(\boldsymbol{\mu_r}, \sum_{\boldsymbol{r}})$, as follows:

$$P\left(l_v|\boldsymbol{\mu_r}, \sum_{\boldsymbol{r}}\right) = \frac{1}{2\pi\sqrt{|\sum_{\boldsymbol{r}}|}} \exp\left(\frac{-(l_v - \boldsymbol{\mu_r})^T \sum_r^{-1}(l_v - \boldsymbol{\mu_r})}{2}\right), \tag{1}$$

where $\boldsymbol{\mu_r}$ and $\boldsymbol{\Sigma_r}$ denote the mean vector and covariance matrix. Note that this component can capture the local preference/attractions within each region. If a POI v is very popular in region r and frequently checked in by users, then the probability of region r generating location l_v is much higher than other locations that receive few check-ins. We apply a multinomial distribution $\boldsymbol{\vartheta_u}$ over regions to model u's spatial patterns.

Modeling the Joint Effect. As a POI has both semantic and geographical attributes, the propensity of a user u for a POI v is determined by the joint effect of u's personal interests and spatial mobility patterns. To model this joint effect, we introduce a joint latent factor topic-region that is responsible for generating the IDs of visited POIs, that is, a topic-region (z, r) is associated with a distribution over POI IDs (i.e., $\boldsymbol{\varphi_{z,r}}$). This joint latent factor also serves to seamlessly unify *user-interest modeling* and *user spatial-pattern modeling*. As a matter of fact, a topic-region represents a geographical area in which POIs have the same or similar semantics (e.g., categories or functions). It comprises two components: semantics and geo-location. For example, POIs in Central Park and those on Wall Street, Manhattan may form two different topic-regions. The ones in Central Park may have semantics such as concert, ticket, bird, running, and so on, while the ones on Wall Street may be associated with the topic of stocks and finances. Meanwhile, the introduction of topic-region enables geographical clustering and topic modeling to influence and enhance each other under a unified framework, since a good geographical division benefits the estimation of topics, and a good topic model helps identify the meaningful geographical segmentation, as analyzed in Yin et al. [2011].

2.3. Generative Process

The generative process of TRM is summarized in Algorithm 1. To avoid overfitting, we place a Dirichlet prior [Blei et al. 2003; Wallach et al. 2009] over each multinomial distribution. Thus, $\boldsymbol{\theta_u}$, $\boldsymbol{\vartheta_u}$, $\boldsymbol{\phi_z}$, $\boldsymbol{\psi_z}$, and $\boldsymbol{\varphi_{z,r}}$ are generated by Dirichlet distributions with parameters α, γ, β, η, and τ, respectively.

Given a user u, the user first selects a topic z according to one's interest distribution $\boldsymbol{\theta}_u$ when planning to visit a POI v. With the chosen topic z, words $\mathcal{W}_v$ are generated from the topic's word distribution $\boldsymbol{\phi}_z$, and time t is generated from the topic's temporal distribution $\boldsymbol{\psi}_z$. Besides the topic, the user also needs to choose a region r according to the user's spatial distribution $\boldsymbol{\vartheta}_u$. With the chosen region r, the POI's geographical coordinate l_v is generated by the region's spatial distribution $\mathcal{N}(\boldsymbol{\mu_r}, \boldsymbol{\Sigma_r})$. Finally, with the chosen topic z and region r, the POI indicator v is generated by the joint topic-region factor $\boldsymbol{\varphi}_{z,r}$, which is a multinomial distribution over POI IDs.

ALGORITHM 1: Probabilistic Generative Process in the TRM

```
for each user $u$ do
    Draw $\boldsymbol{\theta_u} \sim Dirichlet(\cdot|\alpha)$;
    Draw $\boldsymbol{\vartheta_u} \sim Dirichlet(\cdot|\gamma)$;
end
for each topic $z$ do
    Draw $\boldsymbol{\phi_z} \sim Dirichlet(\cdot|\beta)$;
    Draw $\boldsymbol{\psi_z} \sim Dirichlet(\cdot|\eta)$;
end
for each topic $z$ do
    for each region $r$ do
        Draw $\boldsymbol{\varphi_{z,r}} \sim Dirichlet(\cdot|\tau)$;
    end
end
for each $\mathcal{D}_u$ in $\mathcal{D}$ do
    for each check-in $(u, v, l_v, \mathcal{W}_v, t) \in \mathcal{D}_u$ do
        Draw a topic index $z \sim Multi(\boldsymbol{\theta_u})$;
        Draw a time $t \sim Multi(\boldsymbol{\psi_z})$;
        for each token $w \in \mathcal{W}_v$ do
            Draw $w \sim Multi(\boldsymbol{\phi_z})$;
        end
        Draw a region index $r \sim Multi(\boldsymbol{\vartheta}_u)$;
        Draw a location $l_v \sim \mathcal{N}(\boldsymbol{\mu_r}, \boldsymbol{\Sigma_r})$;
        Draw a POI ID $v \sim Multi(\boldsymbol{\varphi_{z,r}})$;
    end
end
```

2.4. Model Inference

Our goal is to learn parameters that maximize the marginal log-likelihood of the observed random variables $\boldsymbol{v}$, $\boldsymbol{l_v}$, $\boldsymbol{\mathcal{W}_v}$, and $\boldsymbol{t}$. However, the exact marginalization is difficult due to the intractable normalizing constant of the posterior distribution. Therefore, we follow the studies by Yin et al. [2014] and Tang et al. [2012] to use collapsed Gibbs sampling for approximate inference, that is, to maximize the complete data likelihood in Equation (2). As a widely used Markov chain Monte Carlo (MCMC) algorithm, Gibbs sampling iteratively samples latent variables (i.e., $\{\boldsymbol{z}, \boldsymbol{r}\}$ in TRM) from a Markov chain, whose stationary distribution is the posterior. The samples can therefore be used to estimate the distributions of interest (i.e., $\{\boldsymbol{\theta}, \boldsymbol{\vartheta}, \boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{\varphi}\}$). For simplicity and speed, we estimate the Gaussian distribution parameters ($\boldsymbol{\mu_r}$, $\boldsymbol{\Sigma_r}$) by the method of moments. As for the hyperparameters α, β, γ, η, and τ, for simplicity, we take fixed values, that is, $\alpha = 50/K$, $\gamma = 50/R$, and $\beta = \eta = \tau = 0.01$, following the studies by Yin et al. [2014] and Tang et al. [2012]. Our algorithm is easily extended to allow these hyperparameters to be sampled and inferred, but this extension can slow down the convergence of the Markov chain.

In the Gibbs sampling procedure, we need to obtain the posterior probabilities of sampling latent topic z and latent region r for each user check-in record $(u, v, l_v, \mathcal{W}_v, t)$, that is, we need to compute the conditional probabilities $P(z|\boldsymbol{z}_\neg, \boldsymbol{r}, \boldsymbol{v}, \boldsymbol{l_v}, \boldsymbol{\mathcal{W}_v}, \boldsymbol{t}, \boldsymbol{u}, \cdot)$ and $P(r|\boldsymbol{z}, \boldsymbol{r}_\neg, \boldsymbol{v}, \boldsymbol{l_v}, \boldsymbol{\mathcal{W}_v}, \boldsymbol{t}, \boldsymbol{u}, \cdot)$, where $\boldsymbol{z}_\neg$ and $\boldsymbol{r}_\neg$ represent topic and region assignments for all check-in records except the current one. According to the Bayes rule, we can compute these conditional probabilities in terms of the joint probability distribution of the latent and observed variables shown in Equation (2). Here, we directly give the sampling formulas, and provide the detailed derivation in Appendix A.

$$
\begin{aligned}
&P(\boldsymbol{v}, \boldsymbol{l_v}, \boldsymbol{\mathcal{W}_v}, \boldsymbol{t}, \boldsymbol{z}, \boldsymbol{r}|\alpha, \beta, \gamma, \tau, \eta, \boldsymbol{\mu}, \boldsymbol{\Sigma})\\
&\quad= P(\boldsymbol{z}|\alpha)P(\boldsymbol{\mathcal{W}_v}|\boldsymbol{z}, \beta)P(\boldsymbol{t}|\boldsymbol{z}, \eta)P(\boldsymbol{r}|\gamma)P(\boldsymbol{v}|\boldsymbol{z}, \boldsymbol{r}, \tau)P(\boldsymbol{l_v}|\boldsymbol{r}, \boldsymbol{\mu}, \boldsymbol{\Sigma})\\
&\quad= P(\boldsymbol{z}|\alpha)\prod_{\boldsymbol{w}\in\boldsymbol{\mathcal{W}_v}} P(\boldsymbol{w}|\boldsymbol{z}, \beta)P(\boldsymbol{t}|\boldsymbol{z}, \eta)P(\boldsymbol{r}|\gamma)P(\boldsymbol{v}|\boldsymbol{z}, \boldsymbol{r}, \tau)P(\boldsymbol{l_v}|\boldsymbol{r}, \boldsymbol{\mu}, \boldsymbol{\Sigma})\\
&\quad= \int P(\boldsymbol{z}|\boldsymbol{\theta})P(\boldsymbol{\theta}|\alpha)d\boldsymbol{\theta}\prod_{\boldsymbol{w}\in\boldsymbol{\mathcal{W}_v}}\int P(\boldsymbol{w}|\boldsymbol{z}, \boldsymbol{\phi})P(\boldsymbol{\phi}|\beta)d\boldsymbol{\phi}\int P(\boldsymbol{t}|\boldsymbol{z}, \boldsymbol{\psi})P(\boldsymbol{\psi}|\eta)d\boldsymbol{\psi}\\
&\qquad\times\int P(\boldsymbol{r}|\boldsymbol{\vartheta})P(\boldsymbol{\vartheta}|\gamma)d\boldsymbol{\vartheta}\int P(\boldsymbol{v}|\boldsymbol{z}, \boldsymbol{r}, \boldsymbol{\varphi})P(\boldsymbol{\varphi}|\tau)d\boldsymbol{\varphi}P(\boldsymbol{l_v}|\boldsymbol{r}, \boldsymbol{\mu}, \boldsymbol{\Sigma})
\end{aligned}
\tag{2}
$$

Sampling topic indicator $\boldsymbol{z}$ for check-in $(u, v, l_v, \mathcal{W}_v, t)$ according to:

$$
\begin{aligned}
P(z|\boldsymbol{z}_\neg, \boldsymbol{r}, \boldsymbol{v}, \boldsymbol{l_v}, \boldsymbol{\mathcal{W}_v}, \boldsymbol{t}, \boldsymbol{u}, \cdot) &\propto \frac{n_{u,z,\neg}+\alpha}{\sum_{z'}(n_{u,z',\neg}+\alpha)}\\
&\times\frac{n_{z,t,\neg}+\eta}{\sum_{t'}(n_{z,t',\neg}+\eta)}\frac{n_{z,r,v,\neg}+\tau}{\sum_{v'}(n_{z,r,v',\neg}+\tau)}\prod_{w\in\mathcal{W}_v}\frac{n_{z,w,\neg}+\beta}{\sum_{w'}(n_{z,w',\neg}+\beta)},
\end{aligned}
\tag{3}
$$

where $n_{u,z}$ is the number of times that latent topic z has been sampled from user u, $n_{z,w}$ is the number of times that word w is generated from topic z, $n_{z,t}$ is the number of times that time slot t is generated from topic z, and $n_{z,r,v}$ is the number of times that POI v is generated from topic-region pair (z, r). The number $n_\neg$ with subscript $\neg$ denotes a quantity excluding the current instance.

Sampling region indicator $\boldsymbol{r}$ for check-in $(u, v, l_v, \mathcal{W}_v, t)$ according to:

$$
\begin{aligned}
&P(r|\boldsymbol{r}_\neg, \boldsymbol{z}, \boldsymbol{v}, \boldsymbol{l_v}, \boldsymbol{\mathcal{W}_v}, \boldsymbol{t}, \boldsymbol{u}, \cdot)\\
&\quad\propto\frac{n_{u,r,\neg}+\gamma}{\sum_{r'}(n_{u,r',\neg}+\gamma)}\frac{n_{z,r,v,\neg}+\tau}{\sum_{v'}(n_{z,r,v',\neg}+\tau)}P(l_v|\boldsymbol{\mu_r}, \boldsymbol{\Sigma_r}),
\end{aligned}
\tag{4}
$$

where $n_{u,r}$ is the number of times that region r has been sampled from user u. After each sampling, we employ the method of moments to update the Gaussian distribution parameters (i.e., $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$) according to the assigned regions $\boldsymbol{r}$ for simplicity and speed. Specifically, parameters $\boldsymbol{\mu_r}$ and $\boldsymbol{\Sigma_r}$ are updated as in Equations (5) and (6):

$$\boldsymbol{\mu_r} = E(r) = \frac{1}{|\mathcal{C}_r|}\sum_{v\in\mathcal{C}_r} l_v \tag{5}$$

$$\boldsymbol{\Sigma_r} = D(r) = \frac{1}{|\mathcal{C}_r|-1}\sum_{v\in\mathcal{C}_r}(l_v-\boldsymbol{\mu_r})(l_v-\boldsymbol{\mu_r})^T, \tag{6}$$

where $\mathcal{C}_r$ denotes the collection of POIs that are associated with the check-ins assigned with latent region r. To speed up the process of Gibbs sampling, we can adopt a late update strategy, that is, to update the model parameters ($\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$) after a full iteration of Gibbs sampler.

Inference Framework. After a sufficient number of sampling iterations, the approximated posteriors can be used to estimate parameters by examining the counts of $\boldsymbol{z}$ and $\boldsymbol{r}$ assignments to check-in records. The detailed inference framework is shown in Algorithm 2. We first initialize the latent geographical regions by a K-means algorithm (Lines 3–4), then randomly initialize the topic assignments for the check-in records (Lines 5–9). Afterwards, in each iteration, Equations (3) and (4) are utilized to update the region and topic assignments for each check-in record $(u, v, l_v, \mathcal{W}_v, t)$ (Lines 14–15). After each sampling, we update the Gaussian distribution parameters (Line 16). The iteration is repeated until convergence (Lines 11–23). In addition, a burn-in process is introduced in the first several hundreds of iterations to remove unreliable sampling results (Lines 19–22). We also introduce the sample lag (i.e., the interval between samples after burn-in) to sample only periodically thereafter to avoid correlations between samples.

Time Complexity. We analyze the time complexity of this inference framework as follows. Suppose that the process needs I iterations to reach convergence. In each iteration, it requires going through all user check-in records. For each check-in record, it first requires $\mathcal{O}(K)$ operations to compute the posterior distribution for sampling a latent topic, then needs $\mathcal{O}(R)$ operations to compute the posterior distribution for sampling a latent region. Thus, the whole time complexity is $\mathcal{O}(I(K+R)D)$, which is linear to the size of the dataset (i.e., the number of check-ins D).

3. ONLINE LEARNING FOR TRM

In practice, users' check-in records are generated in a real-time manner and continually arrive in the form of a data stream. The batch-learning algorithms, such as our developed Algorithm 2, usually suffer from the following two drawbacks when dealing with the situation mentioned earlier: (1) delay of model updates caused by the expensive time cost of rerunning the batch model; and (2) disability to track changing user interests and spatial mobility patterns due to the fact that the latest check-in records used for updating recommendation models are often overwhelmed by the large data of the past. To enable our TRM model to adapt to the check-in streams and support real-time POI recommendations, we develop an online learning model, TRM-Online, using a particle filter.

3.1. Feasibility Analysis

TRM can be viewed as a state space model if we regard the latent variables z and r as hidden state variables, and the check-in record $(u, v, l_v, \mathcal{W}_v, t)$ as an observation variable. A particle filter is a kind of efficient MCMC sampling method for estimating

ALGORITHM 2: Inference Framework of the TRM

```
   Input: User check-in collection $\mathcal{D}$, number of iteration $I$, number of burn-in $I_b$, sample lag
          $I_s$, Priors $\alpha$, $\gamma$, $\beta$, $\eta$, $\tau$
   Output: Estimated parameters $\hat{\boldsymbol{\theta}}$, $\hat{\boldsymbol{\vartheta}}$, $\hat{\boldsymbol{\phi}}$, $\hat{\boldsymbol{\varphi}}$, $\hat{\boldsymbol{\psi}}$, $\hat{\boldsymbol{\mu}}$, and $\hat{\boldsymbol{\Sigma}}$
 1 Create variables $\boldsymbol{\theta}^{sum}$, $\boldsymbol{\vartheta}^{sum}$, $\boldsymbol{\phi}^{sum}$, $\boldsymbol{\varphi}^{sum}$, $\boldsymbol{\psi}^{sum}$, $\boldsymbol{\mu}^{sum}$ and $\boldsymbol{\Sigma}^{sum}$, and initialize them with zero;
 2 Create variables $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$;
 3 Initialize the clustering of geographical locations using K-Means method.
 4 Update $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ according to Equations (5) and (6), respectively;
 5 for each $\mathcal{D}_u \in \mathcal{D}$ do
 6     for each check-in record $(u, v, l_v, \mathcal{W}_v, t) \in \mathcal{D}_u$ do
 7         Assign topic randomly;
 8     end
 9 end
10 Initialize variable count with zero;
11 for iteration $= 1$ to $I$ do
12     for each $\mathcal{D}_u \in \mathcal{D}$ do
13         for each check-in record $(u, v, l_v, \mathcal{W}_v, t) \in \mathcal{D}_u$ do
14             Update topic assignment using Equation (3);
15             Update region assignment using Equation (4);
16             Update $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ according to Equations (5) and (6), respectively;
17         end
18     end
19     if (iteration $> I_b$) and (iteration mod $I_s == 0$) then
20         $count = count + 1$;
21         Update $\boldsymbol{\theta}^{sum}$, $\boldsymbol{\vartheta}^{sum}$, $\boldsymbol{\phi}^{sum}$, $\boldsymbol{\varphi}^{sum}$, $\boldsymbol{\psi}^{sum}$, $\boldsymbol{\mu}^{sum}$ and $\boldsymbol{\Sigma}^{sum}$ as follows:
               $\theta_{u,z}^{sum} += \frac{n_{u,z} + \alpha}{\sum_{z'}(n_{u,z'} + \alpha)}$
               $\vartheta_{u,r}^{sum} += \frac{n_{u,r} + \gamma}{\sum_{r'}(n_{u,r'} + \gamma)}$
               $\phi_{z,w}^{sum} += \frac{n_{z,w} + \beta}{\sum_{w'}(n_{z,w'} + \beta)}$
               $\varphi_{z,r,v}^{sum} += \frac{n_{z,r,v} + \tau}{\sum_{v'}(n_{z,r,v'} + \tau)}$
               $\psi_{z,t}^{sum} += \frac{n_{z,t} + \eta}{\sum_{t'}(n_{z,t'} + \eta)}$
               $\boldsymbol{\mu}_r^{sum} += \boldsymbol{\mu}_r$
               $\boldsymbol{\Sigma}_r^{sum} += \boldsymbol{\Sigma}_r$
22     end
23 end
24 Return model parameters $\hat{\boldsymbol{\theta}} = \frac{\boldsymbol{\theta}^{sum}}{count}$, $\hat{\boldsymbol{\vartheta}} = \frac{\boldsymbol{\vartheta}^{sum}}{count}$, $\hat{\boldsymbol{\phi}} = \frac{\boldsymbol{\phi}^{sum}}{count}$, $\hat{\boldsymbol{\varphi}} = \frac{\boldsymbol{\varphi}^{sum}}{count}$, $\hat{\boldsymbol{\psi}} = \frac{\boldsymbol{\psi}^{sum}}{count}$, $\hat{\boldsymbol{\mu}} = \frac{\boldsymbol{\mu}^{sum}}{count}$, and
   $\hat{\boldsymbol{\Sigma}} = \frac{\boldsymbol{\Sigma}^{sum}}{count}$;
```

parameters of the state space model. Being different from Gibbs sampling, it is an online algorithm.

A particle filter can be applied to solve the inference of the TRM based on the following two reasons. First, suppose that x is the hidden state variable and y is the observation variable in the particle filter. The objective of the particle filter is to estimate the values of hidden state variables given observations, that is, $P(\boldsymbol{x}|\boldsymbol{y})$. On the other hand, our objective in the TRM is to estimate the posterior distribution $P(\boldsymbol{r}, \boldsymbol{z}|\boldsymbol{v}, \boldsymbol{l_v}, \boldsymbol{\mathcal{W}_v}, \boldsymbol{t}, \cdot)$, which is also the probability of state variables given observations. Second, the particle filter assumes that observations are conditionally independent, and observation y_i is only determined by x_i. Obviously, based on the "bag of words" assumption, TRM meets this requirement.

In this article, we employ Rao-Blackwellised particle filtering (RBPF), an enhanced version of particle filtering [Doucet et al. 2000a] as our method. RBPF integrates out the latent variables, thus makes the solutions simpler. In our algorithm, we have P particles; each particle p represents an answer that we desire, that is, the posterior distributions of regions and topics. In our implementation, a particle p stores all region and topic assignments for check-in records together with an important weight $\omega^{(p)}$ that indicates the importance of particle p. A reassignment process is added to enhance the quality of samples.

3.2. Online Learning Algorithm

In this section, we present an online learning algorithm based on RBPF. We divide user check-in records into epochs based on their timestamps to simulate the real-world check-in stream. The epoch length depends on the nature of the application, and can range from a few days to a few weeks. We use $\mathcal{D}^s$ to denote the check-in dataset generated by users at the sth epoch. As shown in Algorithm 3, the overall algorithm consists of two phases: **initialization phase** and **online phase**. The initialization phase accomplishes the task of launching the online phase, while the online phase continually processes the newly arrived check-in records generated by each user and updates the parameter set $\hat{\boldsymbol{\Psi}}$ after processing these records.

In the *initialization phase* (Lines 1–12), for each particle, we apply TRM-Batch on an initial dataset $\mathcal{D}^0$ that contains a small fraction of check-in records. After running over, we get initial region and topic assignments of all initial check-in records, along with sufficient statistics. These values are stored into each particle, which are useful in the online phase.

In the *online phase* (Lines 13–41), we first initialize particle weights with equal values, then process user documents in a check-in stream one after another. Model parameters will be updated every time a check-in is processed. Two new sampling equations are proposed, as follows.

$$P(z^p|\boldsymbol{z}^p_{\neg}, \boldsymbol{r}^p, \boldsymbol{v}, \boldsymbol{l_v}, \boldsymbol{\mathcal{W}_v}, \boldsymbol{t}, \boldsymbol{u}, \cdot) \propto \frac{m^{p,s}_{u,z} + n^{p,s}_{u,z,\neg} + \alpha}{\sum_{z'}(m^{p,s}_{u,z'} + n^{p,s}_{u,z',\neg} + \alpha)} \times \frac{m^{p,s}_{z,t} + n^{p,s}_{z,t,\neg} + \eta}{\sum_{t'}(m^{p,s}_{z,t'} + n^{p,s}_{z,t',\neg} + \eta)} \frac{m^{p,s}_{z,r,v} + n^{p,s}_{z,r,v,\neg} + \tau}{\sum_{v'}(m^{p,s}_{z,r,v'} + n^{p,s}_{z,r,v',\neg} + \tau)} \prod_{w \in \mathcal{W}_v} \frac{m^{p,s}_{z,w} + n^{p,s}_{z,w,\neg} + \beta}{\sum_{w'}(m^{p,s}_{z,w'} + n^{p,s}_{z,w',\neg} + \beta)} \tag{7}$$

$$P(r^p|\boldsymbol{r}^p_{\neg}, \boldsymbol{z}^p, \boldsymbol{v}, \boldsymbol{l_v}, \boldsymbol{\mathcal{W}_v}, \boldsymbol{t}, \boldsymbol{u}, \cdot) \propto \frac{m^{p,s}_{u,r} + n^{p,s}_{u,r,\neg} + \gamma}{\sum_{r'}(m^{p,s}_{u,r'} + n^{p,s}_{u,r',\neg} + \gamma)} \frac{m^{p,s}_{z,r,v} + n^{p,s}_{z,r,v,\neg} + \tau}{\sum_{v'}(m^{p,s}_{z,r,v'} + n^{p,s}_{z,r,v',\neg} + \tau)} P(l_v|\boldsymbol{\mu_r}, \boldsymbol{\Sigma_r})' \tag{8}$$

where $\neg$ has a different meaning from that in Equations (3) and (4). In Equations (3) and (4), it presents *all check-ins* except the current instance, but here it means excluding the current one from the *observed check-ins so far*. The difference is essential between TRM-Batch and TRM-Online. Since we use all currently observed check-in records, including check-ins in the previous epoches, the check-in count in these equations includes two parts. The first part is the contribution of the previous epochs before

ALGORITHM 3: Online Inference of the TRM

```
Input: User check-in collection $\mathcal{D} = \bigcup_s \mathcal{D}^s$;
Output: Estimated parameter set $\hat{\boldsymbol{\Psi}} = \{\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\vartheta}}, \hat{\boldsymbol{\phi}}, \hat{\boldsymbol{\varphi}}, \hat{\boldsymbol{\psi}}, \hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}}\}$;
/* Initialization Phase:                                              */
for $p = 1$ to $P$ do
    while burn-in point is not reached do
        for each $\mathcal{D}_u \in \mathcal{D}^0$ do
            for each check-in record $(u, v, l_v, \mathcal{W}_v, t) \in \mathcal{D}_u$ do
                Draw topic sample $z$ using Equation (3);
                Draw region sample $r$ using Equation (4);
                Update $\boldsymbol{\mu}^p$ and $\boldsymbol{\Sigma}^p$ using Equations (5) and (6), respectively;
            end
        end
    end
end
Calculate sufficient statistics for each particle $p$;
/* Online Phase:                                                      */
Initialize importance weights $\omega^p = 1/P$ for any $p \in \{1, \dots, P\}$;
for $s = 1$ to $S$ do
    for each $\mathcal{D}_u \in \mathcal{D}^s$ do
        for each check-in record $(u, v, l_v, \mathcal{W}_v, t) \in \mathcal{D}_u$ do
            for $p = 1$ to $P$ do
                Draw topic sample $z^p$ using Equation (7);
                Draw region sample $r^p$ using Equation (8);
                Update $\boldsymbol{\mu}^p$ and $\boldsymbol{\Sigma}^p$ using Equations (10) to (13), respectively;
                $\omega^p = \omega^p P(v, l_v, \mathcal{W}_v, t | \boldsymbol{z}_{\neg}^p, \boldsymbol{r}_{\neg}^p, \boldsymbol{v}, \boldsymbol{l_v}, \boldsymbol{\mathcal{W}_v}, \boldsymbol{t}, \boldsymbol{u}, \cdot)$;
            end
            Normalize $\omega^p$ for any $p \in \{1, \dots, P\}$ as $\omega^p = \frac{\omega^p}{\sum_{p'} \omega^{p'}}$;
            Calculate $N_{eff}$ using Equation (15);
            if $N_{eff} \leq N_{thresh}$ then
                Sampling Importance Resample process;
                Randomly select a collection of check-in records $\mathcal{D}(i)$;
                for each check-in $(u, v, l_v, \mathcal{W}_v, t) \in \mathcal{D}(i)$ do
                    for $p = 1$ to $P$ do
                        Draw topic sample $z^p$ using Equation (7);
                        Draw region sample $r^p$ using Equation (8);
                        Update $\boldsymbol{\mu}^p$ and $\boldsymbol{\Sigma}^p$ using Equations (10) to (13), respectively;
                    end
                end
                for $p = 1$ to $P$ do
                    set $\omega^p = 1/P$;
                end
            end
        end
    end
end
Generate parameter set $\hat{\boldsymbol{\Psi}}$;
```

the sth one, denoted as $m^{p,s}$. The second part denotes the contribution of the current check-ins coming in a stream, denoted as $n^{p,s}$. The superscripts p and s in all notations indicate the particle index and epoch index, respectively. For example, $n_{u,z}^{p,s}$ is the number of times that topic z is assigned to user u by the particle p at the sth epoch. Following the work of Ahmed et al. [2011], we use exponential decay with kernel parameter κ defined as follows:

$$m^{p,s} = \sum_{h=0}^{s-1} \exp\left(\frac{h-s}{\kappa}\right) n^{p,h}, \tag{9}$$

since tracking changing patterns such as user interests means forgetting old check-ins quickly. We also tried different functions, such as the linear function, but the exponential decaying achieves the best performance in our experiment.

After each sampling, similar to Equations (5) and (6), we update the model parameters $\boldsymbol{\mu}_{\boldsymbol{r}}^{\boldsymbol{p}}$ and $\boldsymbol{\Sigma}_{\boldsymbol{r}}^{\boldsymbol{p}}$, as follows:

$$\boldsymbol{\mu}_{\boldsymbol{r}}^{\boldsymbol{p,s}} = \frac{1}{|\mathcal{C}_r^{p,s}|} \sum_{v \in \mathcal{C}_r^p} l_v \tag{10}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{r}}^{\boldsymbol{p,s}} = \frac{1}{|\mathcal{C}_r^{p,s}| - 1} \sum_{v \in \mathcal{C}_r^p} (l_v - \boldsymbol{\mu}_{\boldsymbol{r}})(l_v - \boldsymbol{\mu}_{\boldsymbol{r}})^T. \tag{11}$$

Then, they are smoothed by their estimated values in the previous epoches to prevent overfitting, as follows:

$$\boldsymbol{\mu}_{\boldsymbol{r}}^{\boldsymbol{p}} = \frac{\boldsymbol{\mu}_{\boldsymbol{r}}^{\boldsymbol{p,s}} + \exp(\frac{-1}{\kappa})\boldsymbol{\mu}_{\boldsymbol{r}}^{\boldsymbol{p,s-1}}}{1 + \exp(\frac{-1}{\kappa})} \tag{12}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{r}}^{\boldsymbol{p}} = \frac{\boldsymbol{\Sigma}_{\boldsymbol{r}}^{\boldsymbol{p,s}} + \exp(\frac{-1}{\kappa})\boldsymbol{\Sigma}_{\boldsymbol{r}}^{\boldsymbol{p,s-1}}}{1 + \exp(\frac{-1}{\kappa})}. \tag{13}$$

In this algorithm, $P(z^p|\boldsymbol{z}_{\neg}^{p}, \boldsymbol{r}^p, \boldsymbol{v}, \boldsymbol{l_v}, \boldsymbol{\mathcal{W}_v}, \boldsymbol{t}, \boldsymbol{u}, \cdot)$ and $P(r^p|\boldsymbol{r}_{\neg}^{p}, \boldsymbol{z}, \boldsymbol{v}, \boldsymbol{l_v}, \boldsymbol{\mathcal{W}_v}, \boldsymbol{t}, \boldsymbol{u}, \cdot)$ in Equations (7) and (8) are selected as the proposal distributions; thus, the importance weights are updated as in Equation (14), then normalized to sum to 1. $P(v, l_v, \mathcal{W}_v, t|\boldsymbol{z}_{\neg}^{p}, \boldsymbol{r}_{\neg}^{p}, \boldsymbol{v}, \boldsymbol{l_v}, \boldsymbol{\mathcal{W}_v}, \boldsymbol{t}, \boldsymbol{u}, \cdot)$ is the probability of user u generating the current check-in record based on all sampled topic and region assignments so far.

$$\begin{aligned} \omega^p &= \omega^p P(v, l_v, \mathcal{W}_v, t|\boldsymbol{z}_{\neg}^{p}, \boldsymbol{r}_{\neg}^{p}, \boldsymbol{v}, \boldsymbol{l_v}, \boldsymbol{\mathcal{W}_v}, \boldsymbol{t}, \boldsymbol{u}, \cdot) \\ &= \omega^p \sum_z \theta_{u,z}^p \psi_{z,t}^p \prod_{w \in \mathcal{W}_v} \phi_{z,w}^p \sum_r \vartheta_{u,r}^p \varphi_{z,r,v}^p P\left(l_v|\boldsymbol{\mu}_{\boldsymbol{r}}^{\boldsymbol{p}}, \boldsymbol{\Sigma}_{\boldsymbol{r}}^{\boldsymbol{p}}\right), \end{aligned} \tag{14}$$

where $\theta_{u,z}^p$, $\vartheta_{u,r}^p$, $\phi_{z,w}^p$, $\psi_{z,t}^p$ and $\varphi_{z,r,v}^p$ are computed as follows:

$$\begin{aligned} \theta_{u,z}^p &= \frac{m_{u,z}^{p,s} + n_{u,z,\neg}^{p,s} + \alpha}{\sum_{z'}(m_{u,z'}^{p,s} + n_{u,z',\neg}^{p,s} + \alpha)} \\ \vartheta_{u,r}^p &= \frac{m_{u,r}^{p,s} + n_{u,r,\neg}^{p,s} + \gamma}{\sum_{r'}(m_{u,r'}^{p,s} + n_{u,r',\neg}^{p,s} + \gamma)} \\ \phi_{z,w}^p &= \frac{m_{z,w}^{p,s} + n_{z,w,\neg}^{p,s} + \beta}{\sum_{w'}(m_{z,w'}^{p,s} + n_{z,w',\neg}^{p,s} + \beta)} \\ \psi_{z,t}^p &= \frac{m_{z,t}^{p,s} + n_{z,t,\neg}^{p,s} + \eta}{\sum_{t'}(m_{z,t'}^{p,s} + n_{z,t',\neg}^{p,s} + \eta)} \\ \varphi_{z,r,v}^p &= \frac{m_{z,r,v}^{p,s} + n_{z,r,v,\neg}^{p,s} + \tau}{\sum_{v'}(m_{z,r,v'}^{p,s} + n_{z,r,v',\neg}^{p,s} + \tau)}. \end{aligned}$$

Next, effective sample size N_{eff} is calculated as:

$$N_{eff} = 1 \Bigg/ \sum_{p=1}^{P} (\omega^p)^2 \,. \tag{15}$$

N_{eff} measures the efficiency of the method and controls the algorithm to avoid degeneracy [Doucet et al. 2000b]. A sampling importance resample procedure (Line 26) will be run if N_{eff} is no more than the threshold N_{thresh}. P particles are resampled with replacement according to the importance weights. Then, old particles are replaced with the new ones. After this process, particles with small weight have a high possibility to be eliminated. This process reflects the "survival of the fittest" law, that is, "excellent" solutions should be inherited. Intuitively, N_{thresh} decides the frequency of resample, thus influences the effectiveness and speed of the algorithm.

In addition, we add a reassignment process (Lines 28–34) to improve the quality of samples. Since check-ins coming in the online phase are only sampled once, the result might be inaccurate. We solve this problem by picking up some check-ins randomly and reassigning topics and regions to them. $\mathcal{D}_i$ is a collection of randomly selected check-in records whose number is no more than i. For each check-in in $\mathcal{D}_i$, we sample a new topic and a new region according to Equations (7) and (8). Obviously, when $|\mathcal{D}_i|$ is big enough, TRM-Online will degenerate to a batch-learning model, since previous check-ins will be reassigned constantly.

Generally, our final objectives, the posterior distribution $P(\boldsymbol{r}, \boldsymbol{z}|\boldsymbol{v}, \boldsymbol{l}_v, \boldsymbol{\mathcal{W}}_v, \boldsymbol{t}, \cdot)$, as we mentioned in Section 3.1, is approximated as follows:

$$P(\boldsymbol{r}, \boldsymbol{z}|\boldsymbol{v}, \boldsymbol{l}_v, \boldsymbol{\mathcal{W}}_v, \boldsymbol{t}) \approx \sum_{p=1}^{P} \omega^p I(\boldsymbol{z}, \boldsymbol{z}^p) I(\boldsymbol{r}, \boldsymbol{r}^p), \tag{16}$$

where $I(\boldsymbol{z}, \boldsymbol{z}^p)$ and $I(\boldsymbol{r}, \boldsymbol{r}^p)$ are indicator functions:

$$I(\boldsymbol{z}, \boldsymbol{z}^p) = \begin{cases} 1 & \text{if } \boldsymbol{z} \textit{ equals } \boldsymbol{z}^p \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

$$I(\boldsymbol{r}, \boldsymbol{r}^p) = \begin{cases} 1 & \text{if } \boldsymbol{r} \textit{ equals } \boldsymbol{r}^p \\ 0 & \text{otherwise} \end{cases} . \tag{18}$$

In other words, particles with the same vectors $\boldsymbol{z}$ and $\boldsymbol{r}$ have the same $P(\boldsymbol{r}, \boldsymbol{z}|\boldsymbol{v}, \boldsymbol{l}_v, \boldsymbol{\mathcal{W}}_v, \boldsymbol{t})$, and it is equal to the sum of weights of these particles. Then, the optimal topic and region assignments $\boldsymbol{z}^*$ and $\boldsymbol{r}^*$ for parameter estimation are calculated as follows:

$$\boldsymbol{z}^*, \boldsymbol{r}^* = \arg_{\boldsymbol{z}^p, \boldsymbol{r}^p} \max P(\boldsymbol{r}^{\boldsymbol{p}}, \boldsymbol{z}^{\boldsymbol{p}}|\boldsymbol{v}, \boldsymbol{l}_v, \boldsymbol{\mathcal{W}}_v, \boldsymbol{t}). \tag{19}$$

However, in reality, we found that it is very time-consuming to check whether two particles share the same $\boldsymbol{z}$ and $\boldsymbol{r}$, since their length is equal to the size of all check-ins. We also observed that there are seldom the same particles. Therefore, we modify this procedure to choose the particle with the biggest weight for model parameter estimation. The modified $\boldsymbol{z}^*$ and $\boldsymbol{r}^*$ are as follows:

$$\boldsymbol{z}^* = \boldsymbol{z}^{p^*}, \boldsymbol{r}^* = \boldsymbol{r}^{p^*}, p^* = \arg_p \max \omega^p. \tag{20}$$

With $\boldsymbol{z}^*$ and $\boldsymbol{r}^*$, we can easily estimate the model parameters $\hat{\boldsymbol{\Psi}}$ by examining the counts of topic and region assignments to check-in records.

To be able to apply our TRM-Online to large-scale check-in data, we can use a distributed implementation following the architecture in Smola and Narayanamurthy

[2010] and Ahmed et al. [2011]. The state of the sampler comprises the topic-word, topic-time, and topic-region POI count matrixes as well as user-topic and user-region count matrixes. The former are shared across users and are maintained in a distributed hash table using memcached [Smola and Narayanamurthy 2010]. The latter are user-specific and can be maintained locally in each node. We can distribute the users at epoch s across multiple nodes. One key advantage of our model is its ability to process check-in data in an online and distributed fashion, which allows us to process data at a scale.

4. POI RECOMMENDATION USING TRM

Once we have learned the model parameter set $\hat{\mathbf{\Psi}} = \{\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\vartheta}}, \hat{\boldsymbol{\phi}}, \hat{\boldsymbol{\varphi}}, \hat{\boldsymbol{\psi}}, \hat{\boldsymbol{\mu}}, \hat{\mathbf{\Sigma}}\}$, given a target user u_q with the current time t_q and location l_q, that is, $q = (u_q, t_q, l_q)$, we compute a probability of user u_q checking in each unvisited POI v as in Equation (21), then select top-k POIs with the highest probabilities for the target user.

$$P(v|q, \hat{\mathbf{\Psi}}) = \frac{P(v, t_q|u_q, l_q, \hat{\mathbf{\Psi}})}{\sum_{v'} P(v', t_q)|u_q, l_q, \hat{\mathbf{\Psi}})}, \propto P(v, t_q|u_q, l_q, \hat{\mathbf{\Psi}}) \tag{21}$$

where $P(v, t_q|u_q, l_q, \hat{\mathbf{\Psi}})$ is calculated as follows:

$$P(v, t_q|u_q, l_q, \hat{\mathbf{\Psi}}) = \sum_r P(r|l_q, \hat{\mathbf{\Psi}}) P(v, t_q|u_q, r, \hat{\mathbf{\Psi}}), \tag{22}$$

where $P(r|l_q, \hat{\mathbf{\Psi}})$ denotes the probability of user u_q lying in region r given the user's current location l_q, and it is computed as in Equation (23) according to the Bayes rule, in which the prior probability of latent region r can be estimated using Equation (24), as follows.

$$P(r|l_q, \hat{\mathbf{\Psi}}) = \frac{P(r)P(l_q|r, \hat{\mathbf{\Psi}})}{\sum_{r'} P(r')P(l_q|r', \hat{\mathbf{\Psi}})} \propto P(r)P(l_q|r, \hat{\mathbf{\Psi}}) \tag{23}$$

$$P(r) = \sum_u P(r|u)P(u) = \sum_u \frac{N_u + \kappa}{\sum_{u'}(N_{u'} + \kappa)} \hat{\vartheta}_{u',r}, \tag{24}$$

where N_u denotes the number of check-ins generated by user u. In order to avoid overfitting, we introduce the Dirichlet prior parameter κ to play the role of pseudocount. Note that, to support dynamic real-time recommendation, we compute the probability of u_q choosing region r according to the user's real-time location l_q instead of the spatial patterns (i.e., $\vartheta_{u,r}$) learned from the user's historical check-in records, which distinguishes this work from the static recommendation scheme adopted by most POI recommendation work [Ye et al. 2011; Lian et al. 2014; Gao et al. 2013; Hu and Ester 2013; Zhao et al. 2015; Gao et al. 2015].

$P(v, t_q|u_q, r, \hat{\mathbf{\Psi}})$ is computed as in Equation (25), in which we adopt a geometric mean for the probability of topic z generating word set $\mathcal{W}_v$, that is, $P(\mathcal{W}_v|z, \hat{\mathbf{\Psi}}) = \prod_{w \in \mathcal{W}_v} P(w|z, \hat{\mathbf{\Psi}})$, considering that the number of words associated with different POIs may be different.

$$\begin{aligned} P(v, t_q|u_q, r, \hat{\mathbf{\Psi}}) = P(l_v|r, \hat{\mathbf{\Psi}}) \sum_z P(z|u_q, \hat{\mathbf{\Psi}}) P(t_q|z, \hat{\mathbf{\Psi}}) \\ \times \left(\prod_{w \in \mathcal{W}_v} P(w|z, \hat{\mathbf{\Psi}}) \right)^{\frac{1}{W_v}} P(v|z, r, \hat{\mathbf{\Psi}}) \end{aligned}. \tag{25}$$

Based on Equations (22) to (25), the original Equation (21) can be rewritten as in Equation (26).

$$\begin{aligned} P(v|u_q, t_q, l_q, \hat{\boldsymbol{\Psi}}) &\propto \sum_r \Bigg[P(r)P(l_q|r, \hat{\boldsymbol{\Psi}})P(l_v|r, \hat{\boldsymbol{\Psi}}) \sum_z P(z|u, \hat{\boldsymbol{\Psi}})P(t_q|z, \hat{\boldsymbol{\Psi}}) \\ &\quad \times \left(\prod_{w \in \mathcal{W}_v} P(w|z, \hat{\boldsymbol{\Psi}}) \right)^{\frac{1}{W_v}} P(v|z, r, \hat{\boldsymbol{\Psi}}) \Bigg] \\ &= \sum_r \left[P(r)P(l_q|\hat{\boldsymbol{\mu}}_{\boldsymbol{r}}, \hat{\boldsymbol{\Sigma}}_{\boldsymbol{r}})P(l_v|\hat{\boldsymbol{\mu}}_{\boldsymbol{r}}, \hat{\boldsymbol{\Sigma}}_{\boldsymbol{r}}) \sum_z \hat{\theta}_{u_q,z}\hat{\psi}_{z,t_q} \left(\prod_{w \in \mathcal{W}_v} \hat{\phi}_{z,w} \right)^{\frac{1}{W_v}} \hat{\varphi}_{z,r,v} \right] \\ &= \sum_r \sum_z \left[P(r)P(l_q|\hat{\boldsymbol{\mu}}_{\boldsymbol{r}}, \hat{\boldsymbol{\Sigma}}_{\boldsymbol{r}})P(l_v|\hat{\boldsymbol{\mu}}_{\boldsymbol{r}}, \hat{\boldsymbol{\Sigma}}_{\boldsymbol{r}})\hat{\theta}_{u_q,z}\hat{\psi}_{z,t_q} \left(\prod_{w \in \mathcal{W}_v} \hat{\phi}_{z,w} \right)^{\frac{1}{W_v}} \hat{\varphi}_{z,r,v} \right] \end{aligned} \tag{26}$$

$$\begin{aligned} P(v|u_q, t_q, l_q, \hat{\boldsymbol{\Psi}}) &\propto \sum_r [P(r)P(l_q|r, \hat{\boldsymbol{\Psi}})P(l_v|r, \hat{\boldsymbol{\Psi}})] \sum_z \Bigg[P(z|u_q, \hat{\boldsymbol{\Psi}})P(t_q|z, \hat{\boldsymbol{\Psi}}) \\ &\quad \times \left(\prod_{w \in \mathcal{W}_v} P(w|z, \hat{\boldsymbol{\Psi}}) \right)^{\frac{1}{W_v}} \Bigg] \\ &= \sum_r [P(r)P(l_q|\hat{\boldsymbol{\mu}}_{\boldsymbol{r}}, \hat{\boldsymbol{\Sigma}}_{\boldsymbol{r}})P(l_v|\hat{\boldsymbol{\mu}}_{\boldsymbol{r}}, \hat{\boldsymbol{\Sigma}}_{\boldsymbol{r}})] \sum_z \left[\hat{\theta}_{u_q,z}\hat{\psi}_{z,t_q} \left(\prod_{w \in \mathcal{W}_v} \hat{\phi}_{z,w} \right)^{\frac{1}{W_v}} \right] \end{aligned} \tag{27}$$

4.1. Addressing Cold-Start Problem

Cold start is a critical problem in the domain of recommendation. POIs that have been visited by few users or have not been visited by any user are called cold-start POIs. Due to the lack of interaction information between the users and POIs, collaborative-based methods perform poorly. We will show how our TRM models can be applied to the cold-start recommendation scenario. For a cold-start POI v along with its location l_v and content words $\mathcal{W}_v$, it can be recommended to users who may prefer it according to its semantic and geographical attributes. Specifically, the probability of user u checking in a cold-start POI v is computed still according to Equation (21), but the probability $P(v, t_q|u_q, r, \hat{\boldsymbol{\Psi}})$ is redefined as follows:

$$\begin{aligned} &P(v, t_q|u_q, r, \hat{\boldsymbol{\Psi}}) \\ &\quad = P(l_v|r, \hat{\boldsymbol{\Psi}}) \sum_z P(z|u_q, \hat{\boldsymbol{\Psi}})P(t_q|z, \hat{\boldsymbol{\Psi}}) \left(\prod_{w \in \mathcal{W}_v} P(w|z, \hat{\boldsymbol{\Psi}}) \right)^{\frac{1}{W_v}}. \end{aligned}$$

Compared with Equation (25), the probability $P(v|z, r, \hat{\boldsymbol{\Psi}}) = \hat{\varphi}_{z,r,v}$ is not utilized in this equation, since the ID of a cold-start POI v is not available in the training dataset. Thus, for cold-start POI v, the original Equation (21) can be reformulated as in Equation (27), which shows that TRM can effectively alleviate the cold-start problem by leveraging the semantic, temporal, and spatial patterns.

5. EFFICIENT RECOMMENDATION ALGORITHMS

As the time goes on and users' locations are moving, they need different POIs, which requires producing recommendation results in a real-time manner. To support real-time POI recommendation, we develop efficient retrieval algorithms to speed up the process of online recommendation.

5.1. Fast Top-*k* Recommendation Framework

In this section, we first propose a ranking framework according to Equation (26), as follows:

$$\begin{aligned} S(q,v) &= \sum_{a=(z,r)} W(q,a)F(v,a) \\ &= \|\vec{q}\|\|\vec{v}\|\cos(\triangle_{\vec{q},\vec{v}}) \propto \|\vec{v}\|\cos(\triangle_{\vec{q},\vec{v}}) \end{aligned} \tag{28}$$

$$W(q,a) = \hat{\theta}_{u_q,z}\hat{\psi}_{z,t_q}P(l_q|\hat{\boldsymbol{\mu}}_{\boldsymbol{r}}, \hat{\boldsymbol{\Sigma}}_{\boldsymbol{r}}) \tag{29}$$

$$F(v,a) = P(r)P(l_v|\hat{\boldsymbol{\mu}}_{\boldsymbol{r}}, \hat{\boldsymbol{\Sigma}}_{\boldsymbol{r}})\hat{\varphi}_{z,r,v}\left(\prod_{w\in\mathcal{W}_v}\hat{\phi}_{z,w}\right)^{\frac{1}{W_v}}, \tag{30}$$

where $S(q,v)$ represents the ranking score of POI v for query q. Each topic-region pair (z,r) can be seen as an attribute (i.e., $a=(z,r)$); $W(q,a)$ represents the weight of query q on attribute a, and $F(v,a)$ represents the score of POI v with respect to attribute a. We use $\vec{q}$ and $\vec{v}$ to denote the vectors of query q and POI v, respectively, and $\triangle_{\vec{q},\vec{v}}$ is the angle between the two vectors. This ranking framework separates the offline computation from the online computation. Since $F(v,a)$ is independent of queries, it is computed offline. Although the query weight $W(q,a)$ is computed online, its main time-consuming components (i.e., $\hat{\psi}_{z,t_q}$, $\hat{\theta}_{u_q,z}$, and $(\hat{\boldsymbol{\mu}}_{\boldsymbol{r}}, \hat{\boldsymbol{\Sigma}}_{\boldsymbol{r}})$) are also computed offline, the online computation is just a simple combination process. This design enables maximum precomputation for the problem considered and, in turn, minimizes the query time. At query time, the offline scores $F(v,a)$ only need to be aggregated over $A = K\times R$ attributes by a simple weighted sum function.

5.2. Clustering-Based Pruning Algorithm

The straightforward method of generating the top-k recommendation needs to compute the ranking score for all POIs according to Equation (28) and select the k ones with highest ranking scores, which is computationally inefficient, especially when the number of POIs or POI attributes becomes large [Teflioudi et al. 2015; Yin et al. 2015b]. To speed up the online process of producing recommendations, some efficient online recommendation techniques [Koenigstein et al. 2012; Yin et al. 2015b; Teflioudi et al. 2015] have been developed to prune the item (e.g., POI) search space, but they cannot be applied to our problem due to the curse of dimensionality. Specifically, the number of attributes (i.e., $A = K\times R$) is very large, generally beyond 500 dimensions in our problem. The high dimensionality disables existing tree index structures and tree-based search algorithms, such as Metric-Tree [Koenigstein et al. 2012]. Another choice is the threshold algorithm [Yin et al. 2015b; Fagin et al. 2001], which may save the computation for some POIs. However, it also needs to maintain and access a large number of sorted lists and frequently update the *threshold*, which makes it slower. Hashing has been widely utilized to speed up large-scale similarity search (e.g., near neighbor search), but the top-k recommendation problem is not equivalent to similarity search in traditional hashing. For our recommendation problem, queries' preferences over POIs are calculated as the inner product between query and POI features, as shown in Equation (28). The inner product between two vectors is fundamentally different from their cosine similarity unless all POI vectors have the same length, as analyzed in Koenigstein et al. [2012]. Zhou and Zha [2012] applied the conventional hashing methods to speed up the online recommendation, but their method suffers great accuracy loss.

To speed up the online recommendation, we propose a CBB to prune the POI search space and facilitate fast computation of top-k recommendation, inspired by the TA algorithm [Yin et al. 2015b]. It mainly makes use of the simple observation that both the length and the direction of the two vectors $\vec{q}$ and $\vec{v}$ influence the value of their inner product, as shown in Equation (28). Our CBB is also based on another two observations: that (1) a query q only prefers a small number of attributes (i.e., the sparsity of query preferences) and the query weights on most attributes are extremely small; and (2) POIs with high values on these preferred attributes are more likely to become the top-k recommendation results. Our CBB first groups the POI vectors into buckets according to their directions (or angels). To achieve a set of POI clusters (cones) that appropriately fits the distribution of possible POIs, we adopt the spherical K-means algorithm [Dhillon and Modha 2001] to cluster the POI vectors into B buckets. Spherical clustering defines groups of POIs with similar directions or angles. Then, within each bucket $\mathcal{B}$, the POIs are sorted according to their lengths (i.e., $|\vec{v}|$). In addition, for each bucket $\mathcal{B}$, we compute an upper-bound vector $\vec{b}$ whose value on each dimension a is defined as follows:

$$F(b,a) = \max_{v \in \mathcal{B}} F(v,a). \tag{31}$$

The inner product $S(q,b)$ between the upper-bound vector $\vec{b}$ and the query vector $\vec{q}$ represents the upper-bound score for all POIs in $\mathcal{B}$, since the query weight $W(q,a)$ is nonnegative. Note that both clustering POIs into buckets and ranking POIs within each bucket are performed offline.

Given an online query q, we run our CBB, as shown in Algorithm 4, to prune the search space of the POIs, that is, after we have scanned a small number of buckets of POIs, it may not be necessary to examine the remaining buckets. The procedure of this algorithm is as follows. First, we sort the buckets according to the inner product between the query vector $\vec{q}$ and the bucket's upper-bound vector $\vec{b}$ (i.e., $S(q,b)$) (Line 2). Then, we sequentially access the sorted buckets (Lines 3–22). Before examining the POIs in each bucket $\mathcal{B}$, we first check the size of the result list L. When the size of L is no less than k, we will examine the top POI v' in the resulting list (Lines 4–5). We employ a binary min-heap to implement L so that the top POI v' has the kth smallest ranking score in L (Line 5). If its ranking score $S(q,v')$ is no less than the upper-bound score $S(q,b)$, the algorithm terminates early without checking any subsequent buckets (Lines 6–8). Otherwise, we examine each POI v in this bucket $\mathcal{B}$ (Lines 10–21). If the size of L is less than k, we directly add the current POI v with its ranking score to L (Lines 11–13). Otherwise, we compare the ranking score $S(q,v')$ of the top POI v' in L with the score $S(q,v)$ of the current POI v. If $S(q,v')$ is no less than $S(q,v)$, we skip the current POI. Otherwise, v' in L is replaced by the current POI v (Lines 16–19).

ALGORITHM 4: Clustering-based Branch and Bound Algorithm

```
Input: A POI set $V$, a query $q$ and $B$ buckets;
Output: Result list $L$ with $k$ highest-ranking scores;
1  Initialize $L$ as $\emptyset$;
2  Sort the buckets according to $S(q, b)$;
3  for $i = 1$ to $B$ do
4      if $L.size() \geq k$ then
5          $v' = L.top()$;
6          if $S(q, v') \geq S(q, b_i)$ then
7              break;
8          end
9      end
       /* POIs in each bucket $\mathcal{B}_i$ are sorted according to their lengths.   */
10     for POI $v \in \mathcal{B}_i$ do
11         if $L.size() < k$ then
12             $L.add(v, S(q, v))$;
13         end
14         else
15             $v' = L.top()$;
16             if $S(q, v) > S(q, v')$ then
17                 $L.removeTop()$;
18                 $L.add(v, S(q, v))$;
19             end
20         end
21     end
22 end
23 $L.Reverse()$;
24 Return $L$;
```

Discussion. It is easy to understand that Algorithm 4 is able to correctly find the top-k POIs with the monotone aggregation function $S(q, v)$ defined in Equation (28). Next, we will prove it formally.

THEOREM 1. *Algorithm 4 is able to correctly find the top-k POIs with the monotone aggregation function $S(q, v)$ defined in Equation (28).*

PROOF. Let L be a ranked list returned by Algorithm 4, which contains the k POIs that have been seen with the highest-ranking scores. We only need to show that every POI in L has a ranking score at least as high as any other item v not in L. By definition of L, this is the case for each item v that has been examined in running Algorithm 4. Thus, assume that v was not examined, and the score of v on each attribute a is $F(v, a)$. $\vec{b}$ is assumed to be the upper-bound vector for the bucket into which v is clustered. By definition of $\vec{b}$ in Equation (31), $F(v, a) \leq F(b, a)$, for every a. Hence, the inequality $S(q, v) \leq S(q, b)$ holds because of the monotonicity of the ranking function defined in Equation (28). But, by definition of L, for every v' in L, we have that $S(q, v') \geq S(q, b)$. Therefore, for every v' in L, we have that $S(q, v') \geq S(q, b) \geq S(q, v)$, as desired. □

Compared with the classic TA retrieval algorithm developed in Yin et al. [2015b] and Fagin et al. [2001], our proposed CBB algorithm need not dynamically maintain a priority queue of sorted lists, nor frequently update a threshold score. Therefore, our proposed CBB is more efficient than TA, especially when the data dimensionality is very high (more than 500), as shown in Section 6.8.

Table III. Basic Statistics of Foursquare and Twitter Datasets

	Foursquare	Twitter
# of users	4,163	114,508
# of POIs	21,142	62,462
# of total check-ins	483,813	1,434,668
# of hometown check-ins	445,107	1,408,886
# of out-of-town check-ins	38,706	25,782
time span	Dec 2009–Jul 2013	Sep 2010–Jan 2011

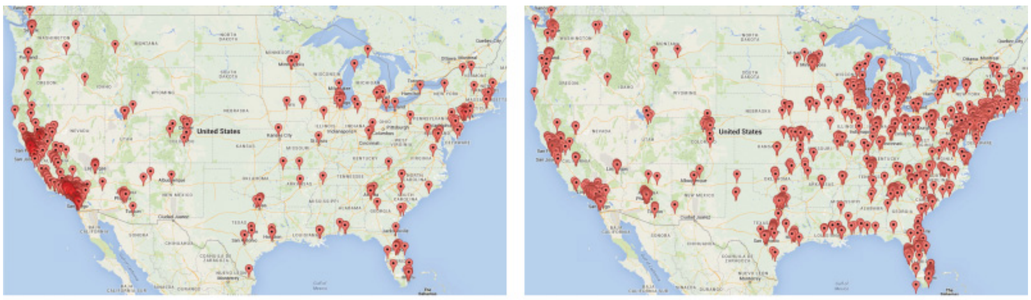

(a) Foursquare (b) Twitter

Fig. 2. Distribution of user check-in activities.

6. EXPERIMENTS

In this section, we first describe experimental settings including datasets, comparative approaches, and evaluation methods. Then, we demonstrate the experimental results on recommendation effectiveness and efficiency as well as model-training efficiency.

6.1. Datasets

Following the method developed in Gao et al. [2012], we collected two large-scale real-life datasets to conduct experiments: Foursquare and Twitter. Since we focus on new POI recommendation, we removed the repeated check-ins from the two datasets. Their basic statistics are shown in Table III.

Foursquare. This dataset contains the check-in history of 4,163 users who live in California. For each user, it contains the user's social networks, check-in POI IDs, location of each check-in POI in terms of latitude and longitude, check-in time and the contents of each check-in POI. The total number of check-in records in this dataset is 483,813. Each check-in record is stored as *user-ID, POI-ID, POI-location, POI-contents, check-in time*. Each record in social networks is stored as *userID, friendID*, and the total number of social relationships is 32,512. All users in this dataset are active users who have at least 10 check-ins. The distribution of the check-in activities is described in Figure 2(a). We can see from the distribution that, although most of the check-ins are located in California for users who live in California, a number of check-ins nevertheless occur in other states. This is sound proof of the significance of out-of-town recommendations.

Twitter. This dataset is based on the publicly available Twitter dataset [Cheng et al. 2011]. Twitter supports third-party location-sharing services such as Foursquare and Gowalla (in which users of these services can share their check-ins on Twitter). However, the original dataset does not contain the category and tag information about each POI. Thus, we crawled the category and tag information associated with each POI from

Foursquare with the help of its publicly available API[2]. The enhanced dataset contains 114,058 users and 1434,668 check-ins. Each check-in record has the same format with the Foursquare dataset provided here; however, this dataset does not contain user social network information. Figure 2(b) illustrates the distribution of the check-in activities across the United States. In this dataset, 18.22% of check-in activities are located in California, and fewer than 7% of activity records remain in each of the other states.

Note that users' home locations are not explicitly available in these two datasets. Thus, we employ the widely adopted method in Scellato et al. [2011], Cho et al. [2011], and Yin et al. [2015] which discretizes the world into 25*km*-by-25*km* cells and defines the home location as the average position of check-ins in the cell with most of the user's check-ins. To make the experiments repeatable, we make the two datasets publicly available[3].

6.2. Comparative Approaches

Recommendation Effectiveness. We first compare our TRM models (TRM-Batch and TRM-Online) with the following four competitive methods that represent state-of-the-art POI recommendation techniques.

SVDFeature. SVDFeature [Chen et al. 2012] is a machine-learning toolkit designed to solve the feature-based matrix factorization. Based on this toolkit, we build a factorization model incorporating more side information beyond the user-POI matrix, including POI content, POI geographical location and temporal dynamics (i.e., check-in time), to compare with our models fairly. The limitation of SVDFeature is that it cannot deal with continuous time and location; thus, we adopt the discretization methods developed in Yuan et al. [2013] and Yin et al. [2015a] to segment them into bins and grid squares.

Category-based KNN (CKNN). CKNN, developed in Bao et al. [2012], first projects a user's activity history into a well-designed category space and models each user's preferences with a weighted category hierarchy. Meanwhile, it infers the authority of each user in a city using the HITS model. When receiving a query q, CKNN first selects a set of users who have both high authority at the target city and similar preferences with the querying user u; then, recommendations are produced according to check-in records of these selected users.

LCA-LDA. LCA-LDA is a location-content-aware recommender model developed to support POI recommendations for users traveling in new cities [Yin et al. 2014]. This model takes into account both personal interests and local preferences of each city by exploiting both POI co-visiting patterns and *semantic information*. Compared with the TRM, it does not consider the temporal and spatial patterns of users' check-in activities.

UPS-CF. UPS-CF, proposed in Ference et al. [2013], is a collaborative recommendation framework that incorporates social influence to support out-of-town recommendations. This framework integrates user-based collaborative filtering and social-based collaborative filtering, that is, to recommend POIs to a target user according to the check-in records of both the user's friends and similar users.

Note that there are many other existing POI recommendation techniques, but they cannot support out-of-town recommendations, as analyzed in Section 7. Therefore, we do not compare with them. In the experiment, we first use a validation set to find the optimal hyperparameter setups for these recommendation models by performing a grid search, such as the number of topics (K) and the number of regions (R) in the TRM.

To further validate the benefits brought by exploiting the semantic, temporal, and spatial patterns, respectively, we design three variant versions. **TRM-S1** is the first

[2]https://developer.foursquare.com/.
[3]https://sites.google.com/site/dbhongzhi/.

simplified version of the TRM, in which we remove content information of check-in POIs, and the latent variable z generates only the check-in time t for each check-in record. In **TRM-S2**, we remove the check-in time information, and the latent variable z generates only the word set $\mathcal{W}_v$ for each check-in record. As the third simplified version, **TRM-S3** does not consider the geographical information of check-in POIs, and the latent variable r is thus removed.

Recommendation Efficiency. We compare our CBB algorithm with two competitor algorithms and one baseline. The first competitor, developed in Koenigstein et al. [2012] to speed up the online recommendation, first uses a binary spatial partitioning metric tree (**MT**) to index the POI vectors, then utilizes a branch-and-bound algorithm to prune the POI search space. The second competitor is the threshold algorithm (**TA**) [Yin et al. 2015b] developed for online recommendation. It precomputes a sorted list for each latent attribute a in which POIs are sorted according to their values on attribute a, and maintains a priority queue of the A sorted lists that controls which sorted list to access in the next. The algorithm has the nice property of terminating early without scanning all POIs. The baseline is a brute-force (**BF**) algorithm that needs to compute a ranking score for each POI and selects top-k ones with highest-ranking scores.

6.3. Evaluation Methods

Recommendation Effectiveness. To make the evaluation process fair and reproducible, we adopt the *methodological description framework* proposed in Campos et al. [2014] and Yin et al. [2015b] to describe our evaluation conditions. We will present our evaluation conditions by answering the following methodological questions:

(1) What *base set* is used to perform the training-test building?
(2) What *rating order* is used to assign ratings to the training and test sets?
(3) How many *ratings* comprise the training and test sets?
(4) Which items are considered as *target items*?

Base set condition. The base set conditions state whether the splitting procedure of training and test sets is based on the whole collection of check-ins $\mathcal{D}$ or on each of the subdatasets of $\mathcal{D}$ independently. We adopt the *user-centered base set condition*. Specifically, for each user u, the user's check-in records $\mathcal{D}_u$ are first divided into hometown check-ins $\mathcal{D}_u^{home}$ and out-of-town check-ins $\mathcal{D}_u^{out}$, since our the TRM is designed for both hometown and out-of-town recommendations, and we need to evaluate the recommendation effectiveness under the two scenarios. To decide whether a check-in record occurs at hometown or out of town, we measure the distance between the user's home location and the POI (i.e., $|l_u - l_v|$). If the distance is less than d, then we assume that the check-in occurs at hometown. Otherwise, the check-in record is assumed to be generated out of town. Note that the distribution of check-ins generated at the hometown and out-of-town settings is highly imbalanced, as shown in Table III. Then, we perform the splitting independently on each user's hometown check-in document $\mathcal{D}_u^{home}$ and out-of-town check-in document $\mathcal{D}_u^{out}$, ensuring that all users will have check-ins in both the training and test sets for the two recommendation scenarios.

Rating order and size conditions. To simulate a more real recommendation scenario, we adopt the *time-dependent rating order condition*. The check-in records in both $\mathcal{D}_u^{home}$ and $\mathcal{D}_u^{out}$ are first ranked according to their check-in timestamps. Then, we use the 80th percentile as the cut-off point so that check-ins before this point will be used for training and the rest are for testing. In the training dataset, we choose the last 10% check-ins as the validation data to tune the model hyperparameters such as the numbers of topics and regions (i.e., K and R).

Target item condition. To simulate a real-world setting, given a target user u and a ground truth POI v, we require each tested recommender system to rank all the

POIs within the circle of radius d centered at l_v except those in the user's training set, instead of all available POIs, since only those POIs that are geographically close to v are comparable with v. This design can effectively simulate the local competition effect and user behavior of choices.

This condition combination is also called "uc_td_prop" for short. According to these evaluation conditions, the whole dataset $\mathcal{D}$ is split into the training set $\mathcal{D}_{train}$ and the test set $\mathcal{D}_{test}$. To simulate a more real POI recommendation scenario, especially out-of-town recommendation, we have to choose a location coordinate as the target user's current standing position before visiting v. Specifically, for each test case $(u, v, l_v, W_v, t) \in \mathcal{D}_{test}$, we use a Gaussian function with the center l_v to generate a geographical coordinate l to represent the current standing point of user u. Thus, a query $q = (u, l, t)$ is formed for the test case.

To evaluate the recommendation methods, we adopt the evaluation methodology and measurement Accuracy@k proposed in Hu and Ester [2013] and Yin et al. [2014]. Specifically, for each check-in $(u, v, l_v, \mathcal{W}_v, t)$ in $\mathcal{D}_{test}$:

(1) We compute the ranking score for POI v and all other POIs that are within the circle of radius d centered at l_v and unvisited by u previously.
(2) We form a ranked list by ordering all of these POIs according to their ranking scores. Let p denote the position of the POI v within this list. The best result corresponds to the case in which v precedes all the unvisited POIs (i.e., $p = 1$).
(3) We form a top-k recommendation list by picking the k top-ranked POIs from the list. If $p \leq k$, we have a hit (i.e., the ground truth v is recommended to the user). Otherwise, we have a miss.

The computation of Accuracy@k proceeds as follows. We define hit@k for a single test case as either the value 1, if the ground truth POI v appears in the top-k results, or the value 0, if otherwise. The overall Accuracy@k is defined by averaging over all test cases:

$$Accuracy@k = \frac{\#hit@k}{|\mathcal{D}_{test}|},$$

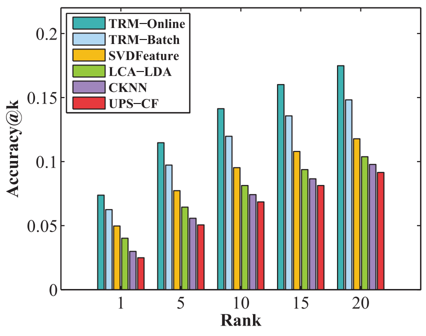
where $\#hit@k$ denotes the number of hits in the test set, and $|\mathcal{D}_{test}|$ is the number of all test cases.

Recommendation Efficiency. The efficiency of the online recommendations mainly depends on (1) the number of available POIs in the dataset, (2) the number of POIs required to be recommended (k) and (3) the number of POI attributes (A). Therefore, we test the recommendation efficiency over these three factors.

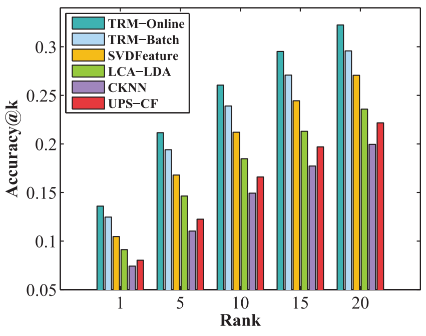
6.4. Recommendation Effectiveness

In this section, we report the comparison results between our proposed TRM models and other competitive methods with well-tuned parameters. Figures 3 and 4 report the performance of the recommendation methods on the Foursquare and Twitter datasets, respectively. It is apparent that these different recommendation methods have significant performance disparity in terms of top-k accuracy. We only show the performance for which k is set to 1, 5, 10, 15, and 20, as a greater value of k is usually ignored for a typical top-k recommendation task.

Out-of-Town Recommendations on Foursquare. Figure 3(a) presents the recommendation accuracy in the scenario of out-of-town recommendations, for which the accuracy of TRM-Online is about 0.141 when $k = 10$, and 0.174 when $k = 20$ (i.e., the model has a probability of 14.1% of placing an appealing POI in the top 10 and 17.4% of placing it in the top 20). Clearly, our proposed TRM models outperform other competitive models significantly, and the advantages of TRM models over other competitive

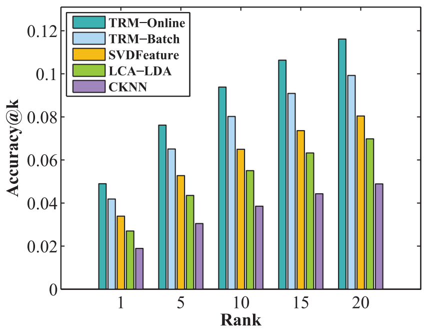


(a) Out-of-town Recommendation

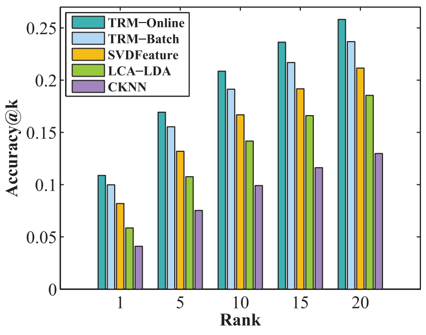


(b) Home-town Recommendation

Fig. 3. Top-k performance on Foursquare dataset.



(a) Out-of-town Recommendation



(b) Home-town Recommendation

Fig. 4. Top-k performance on the Twitter dataset.

methods are very obvious in this scenario, showing the benefits of jointly exploiting the semantic patterns, temporal patterns, and geographical patterns of users' check-in activities. Several observations are made from the results: (1) UPS-CF drops behind the other five methods, showing the advantages of exploiting the content information of users' visited POIs to capture their interests. Through the medium of content, TRM-Online, TRM-Batch, LCA-LDA, CKNN, and SVDFeature transfer the users' interests inferred at hometown to out-of-town regions. In contrast, UPS-CF is a mixture of collaborative filtering and social filtering, which ignores the effect of content. (2) TRM models achieve much higher accuracy than LCA-LDA and CKNN, showing the benefits of considering both temporal and geographical patterns. (3) TRM models perform much better than SVDFeature, although they use the same types of features and information, showing the advantage of the well-designed probabilistic generative model incorporating domain knowledge over the general feature-based matrix factorization. (4) TRM-Online performs better than TRM-Batch, because TRM-Online can capture the dynamics of user interests while TRM-Batch assumes that users' interests $\boldsymbol{\theta}_u$ are stable. That is, changes in user interests are not taken into consideration in TRM-Batch. However, users' interests are not always stable and may change as time goes by or they move from a city to another. For instance, users will naturally be interested in parenting venues after they have a baby. For another, when a Google employee transfers from Beijing, China to Mountain View in the United States, that employee's interests will most likely change. In our TRM-Online model, a POI that was visited

Table IV. Recommendation Accuracy on Foursquare Dataset

Methods	Out-of-Town Scenario			Hometown Scenario		
	Ac@1	Ac@10	Ac@20	Ac@1	Ac@10	Ac@20
TRM-Batch	**0.062**	**0.119**	**0.148**	**0.124**	**0.239**	**0.295**
TRM-Batch-S1	0.049	0.095	0.117	0.117	0.225	0.278
TRM-Batch-S2	0.055	0.107	0.132	0.106	0.203	0.252
TRM-Batch-S3	0.059	0.113	0.140	0.110	0.210	0.261
TRM-Online	**0.073**	**0.141**	**0.174**	**0.136**	**0.261**	**0.322**
TRM-Online-S1	0.058	0.112	0.138	0.128	0.245	0.303
TRM-Online-S2	0.065	0.126	0.156	0.115	0.222	0.274
TRM-Online-S3	0.069	0.133	0.165	0.119	0.229	0.284

recently by a user has a bigger impact on the prediction of the user's future visiting behaviors than a POI that was visited a long time ago.

Hometown Recommendations on Foursquare. In Figure 3(b), we report the performance of all recommendation models for the hometown scenario; our TRM-Online achieves the highest recommendation accuracy. From the results, we observe that the recommendation accuracies of all methods are higher in Figure 3(b) than that in Figure 3(a). In addition, CKNN outperforms UPS-CF in Figure 3(a), while UPS-CF slightly exceeds CKNN in Figure 3(b), showing that the collaborative filtering better suits the hometown recommendation setting, in which the user-POI matrix is not sparse, and the content-based filtering such as CKNN is more capable of overcoming the issue of data sparsity in the out-of-town scenario. TRM, SVDFeature and LCA-LDA are hybrid recommendation methods that take advantage of different dimension information; thus, they perform well consistently in both recommendation settings. The comparison between Figure 3(a) and Figure 3(b) also reveals that the two recommendation scenarios are intrinsically different, and should be separately evaluated.

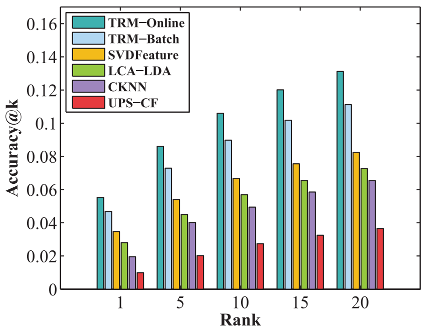
Recommendation on Twitter. Figure 4 reports the performance of the recommendation models on the Twitter dataset. We do not compare our models with UPS-CF since this dataset does not contain user social network information. From the figure, we can see that the trend of comparison result is similar to that presented in Figure 3; the main difference is that all recommendation methods achieve lower accuracy. This may be because users in the Foursquare dataset have more check-in records than users in the Twitter dataset, on average, which enables the models to capture users' interests and preferences more accurately.

6.5. Impact of Different Factors

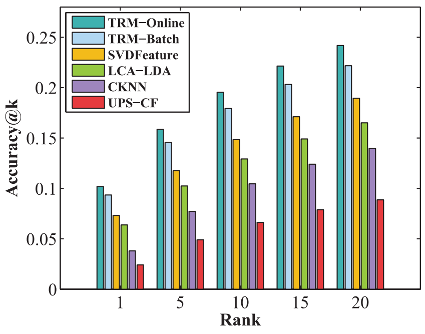
In this section, we carry out an ablation study showing the benefits of exploiting semantic, temporal, and spatial patterns, respectively. We compare our TRM models (TRM-Batch and TRM-Online) with three variant versions, TRM-S1, TRM-S2, and TRM-S3. The comparison results are shown in Tables IV and V. From the results, we first observe that TRM models consistently outperform the three variants in both out-of-town and hometown recommendation scenarios, indicating that our TRM models benefit from simultaneously considering the three factors and their joint effect on users' decision making. Second, we observe that the contribution of each factor to improving recommendation accuracy is different. Another observation is that the contributions of the same factor are different in the two different recommendation scenarios. Specifically, according to the importance of the three factors in the out-of-town recommendation scenario, they can be ranked as Semantic Patterns > Temporal Patterns > Spatial Patterns, while in the hometown recommendation scenario, they can be ranked as Temporal Patterns > Spatial Patterns > Semantic Patterns. Obviously,

Table V. Recommendation Accuracy on Twitter Dataset

Methods	Out-of-Town Scenario			Hometown Scenario		
	Ac@1	Ac@10	Ac@20	Ac@1	Ac@10	Ac@20
TRM-Batch	**0.041**	**0.081**	**0.101**	**0.099**	**0.191**	**0.236**
TRM-Batch-S1	0.033	0.063	0.078	0.094	0.180	0.223
TRM-Batch-S2	0.037	0.071	0.088	0.085	0.163	0.201
TRM-Batch-S3	0.039	0.075	0.093	0.088	0.168	0.209
TRM-Online	**0.049**	**0.094**	**0.117**	**0.108**	**0.208**	**0.258**
TRM-Online-S1	0.039	0.075	0.093	0.102	0.196	0.243
TRM-Online-S2	0.044	0.084	0.104	0.092	0.177	0.220
TRM-Online-S3	0.046	0.089	0.110	0.096	0.184	0.227



(a) Out-of-town Recommendation



(b) Home-town Recommendation

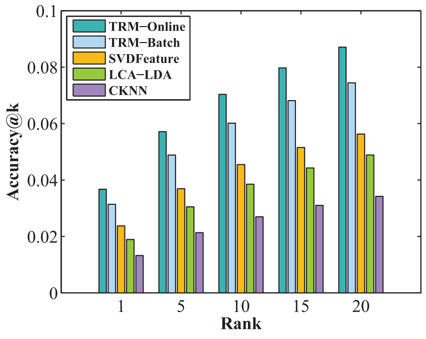
Fig. 5. Recommendation accuracy for cold-start POIs on Foursquare dataset.

the semantic patterns play a dominant role in overcoming the issue of data sparsity in the out-of-town recommendation scenario, while the temporal cyclic patterns are most important to improve hometown recommendations. This is because the two recommendation scenarios have different characteristics: (1) most users have enough check-in records in their hometowns while few check-in activities are left in out-of-town regions; (2) the limitation of travel distance in the out-of-town scenario does not matter as much as that in the hometown scenario; and (3) users' daily routines may change when they travel out of town.

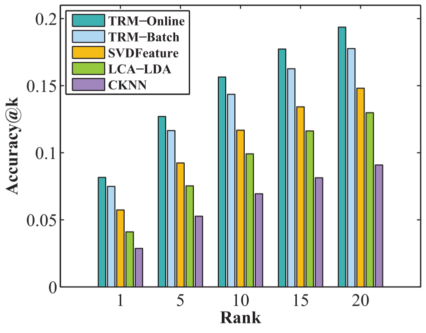
6.6. Test for Cold-Start Problem

We further conduct experiments to study the effectiveness of different recommendation algorithms handling the cold-start problem. We test the recommendation effectiveness for cold-start POIs on both Foursquare and Twitter datasets, and present the results in Figures 5 and 6. We define those POIs that are visited by less than 5 users as cold-start POIs. To test the performance of cold-start POI recommendations, we select users who have at least one cold-start check-in as test users. For each test user, we first choose the user's check-in records associated with cold-start POIs as test data $\mathcal{D}_{test}$ and the remaining check-in records as training data $\mathcal{D}_{training}$. We aim to measure whether the marked-off cold-start POIs in $\mathcal{D}_{test}$ can be accurately recommended to the right users in the top-k results.

From the results, we have the following observations: (1) our proposed TRM models perform best consistently in recommending cold-start POIs; and (2) compared with the results in Figures 3 and 4, the recommendation accuracy of all algorithms decreases, to different degrees, for cold-start POIs, for example, the recommendation accuracy



(a) Out-of-town Recommendation



(b) Home-town Recommendation

Fig. 6. Recommendation accuracy for cold-start POIs on Twitter dataset.

Table VI. Out-of-Town Recommendation Accuracy@10

R \ K	K=10	K=20	K=30	K=40	K=50	K=60
R=5	0.109	0.117	0.122	0.124	0.124	0.125
R=10	0.117	0.125	0.130	0.133	0.133	0.133
R=15	0.121	0.129	0.134	0.137	0.137	0.137
R=20	0.124	0.133	0.138	**0.141**	0.141	0.141
R=25	0.124	0.133	0.139	0.141	0.141	0.142
R=30	0.125	0.133	0.139	0.141	0.141	0.142

of UPS-CF drops drastically, while our TRM models deteriorate slightly; (3) all TRM, SVDFeature, LCA-LDA, and CKNN perform significantly better than UPS-CF in both out-of-town and hometown recommendation scenarios, which demonstrates that the recommendation methods that incorporate semantic contents of POIs perform significantly better than both collaborative-filtering and social-filtering methods. This is because cold-start POIs lack interaction information, which is the essential foundation of both collaborative-filtering and social-filtering methods. The superior performance of TRM models in recommending cold-start POIs shows that exploiting and integrating semantic, temporal, and spatial patterns can effectively alleviate the cold-start problem.

6.7. Parameter Sensitivity Analysis

Tuning model parameters—such as the number of topics (i.e., K), the number of regions (i.e., R) and the number of particles—is critical to the performance of our TRM models. We therefore study the effect of tuning model parameters on the Foursquare dataset.

As for the hyperparameters α, β, γ, and η for simplicity, we take a fixed value, that is, $\alpha = 50/K$, $\gamma = 50/R$, and $\beta = \eta = \tau = 0.01$, following the studies by Yin et al. [2014] and Tang et al. [2012]. We try different setups and find that the performance of TRM models is not sensitive to these hyperparameters, but their performance is sensitive to the number of topics and regions. Thus, we test the performance of the TRM-Online model by varying the number of topics and regions. The results are presented in Tables VI and VII. The results are consistent for the two different recommendation scenarios. From the results, we observe that the recommendation accuracy of TRM-Online first increases with the increasing number of topics, then it does not change significantly when the number of topics is larger than 40. A similar observation is made for increasing the number of regions (i.e., R): the recommendation accuracy of TRM-Online increases with the increasing number of regions, then it does not change much

Table VII. Hometown Recommendation Accuracy@10

R \ K	K=10	K=20	K=30	K=40	K=50	K=60
R=5	0.202	0.215	0.225	0.229	0.229	0.230
R=10	0.215	0.230	0.240	0.245	0.245	0.245
R=15	0.222	0.238	0.248	0.253	0.253	0.253
R=20	0.229	0.245	0.255	**0.261**	0.261	0.261
R=25	0.229	0.245	0.256	0.261	0.261	0.261
R=30	0.230	0.245	0.256	0.261	0.261	0.261

Table VIII. Out-of-Town Recommendation Accuracy@10

P \ $\lvert\mathcal{D}(i)\rvert$	10	20	40	60	80	100	200
10	0.107	0.116	0.119	0.121	0.121	0.119	0.118
20	0.119	0.130	0.133	0.135	0.135	0.133	0.131
30	0.124	0.135	0.138	0.141	0.141	0.138	0.137
40	0.124	0.136	0.138	0.141	0.141	0.138	0.137
50	0.124	0.136	0.139	0.141	0.141	0.139	0.137
60	0.125	0.136	0.139	0.142	0.142	0.139	0.137

Table IX. Home-town Recommendation Accuracy@10

P \ $\lvert\mathcal{D}(i)\rvert$	10	20	40	60	80	100	200
10	0.197	0.215	0.220	0.224	0.224	0.220	0.217
20	0.220	0.240	0.245	0.250	0.250	0.245	0.243
30	0.229	0.250	0.255	0.261	0.261	0.255	0.253
40	0.230	0.251	0.256	0.261	0.261	0.256	0.253
50	0.230	0.251	0.256	0.261	0.261	0.256	0.253
60	0.230	0.251	0.256	0.262	0.262	0.256	0.254

when the number of regions is larger than 20. The reason is that K and R represent the model complexity. Thus, when K and R are too small, the model has a limited ability to describe the data. On the other hand, when K and R exceed a threshold, the model is expressive enough to handle the data. At this point, it is less helpful to improve the model performance by increasing K and R. Meanwhile, the time cost of training our model increases with the increasing K and R. Similar observations are also made for the TRM-Batch model. As a trade-off between the model effectiveness and the model efficiency, the performance reported in Figure 3 is achieved with 40 latent topics (i.e., $K = 40$) and 20 latent regions (i.e., $R = 20$). As the Twitter dataset is more widespread in the geographical space and contains more POI categories than the Foursquare dataset, the experimental results presented in Figure 4 are obtained with the parameter settings $K = 50$ and $R = 30$.

As for the parameters $|\mathcal{D}(i)|$, N_{thresh}, and P, which influence the effectiveness and runtime of TRM-Online, we should make a trade-off. Following many recent empirical studies [Doucet et al. 2000a], N_{thresh} is set to 1.5 since the particle filtering-based online learning algorithms can achieve a good trade-off for $N_{thresh} = 1.5$. In the following experiment, we test the performance of the TRM-Online model by varying P and $|\mathcal{D}(i)|$ and show the results in Tables VIII and IX. The results are consistent for the two recommendation scenarios. Initially, increasing P has a big improvement on the results, but this effect quickly slows ($P > 30$). These results demonstrate that we are able to learn a high-performance TRM model using only a small number of particles. Another observation is that, as $|\mathcal{D}(i)|$ increases, the recommendation accuracy of the

Table X. Recommendation Accuracy@10

Settings \ Distance (km)	d=10	d=50	d=100	d=200	d=300	d=500
Out-of-Town Performance	0.223	0.165	0.141	0.141	0.141	0.141
Hometown Performance	0.262	0.262	0.261	0.245	0.244	0.244

Table XI. Recommendation Efficiency on Twitter Dataset

Methods	Online Recommendation Time Cost (ms)				
	k=1	k=5	k=10	k=15	k=20
CBB	**53.91**	**65.75**	**71.57**	**74.13**	**78.32**
BF	110.83	111.40	111.81	111.95	112.94
MT	130.90	131.46	131.87	132.39	132.99
TA	880.92	1055.68	1144.82	1221.49	1265.58

TRM-Online model first increases, then decreases. One possible reason for early increasing performance is that check-ins arriving in the form of stream are only sampled once; thus, the sample results might be inaccurate. The quality of samples is improved by adding a reassignment process for randomly selected check-ins $\mathcal{D}(i)$. Later, the recommendation accuracy decreases as $|\mathcal{D}(i)|$ gets larger, because TRM-Online degenerates to a batch-learning model that is incapable of tracking changing user interests when the number of check-ins in $\mathcal{D}(i)$ is large enough. TRM-Online achieves its best performance when $|\mathcal{D}(i)|$ is set to 60 on this dataset, which could be a trade-off of the two factors discussed earlier. Similar observations are made on the Twitter dataset, and the optimal experimental results presented in Figure 4 are obtained with the same parameter setup as on the Foursquare dataset (i.e., $P = 40$ and $|\mathcal{D}(i)| = 60$).

We also study the impact of the distance d on the Foursquare dataset and show the results in Table X, since it is a critical parameter to control whether a check-in happens at hometown or out-of-town. From the results, we observe that the out-of-town recommendation accuracy first decreases significantly with the increasing distance d, then it does not change when d is larger than 100km; while the hometown recommendation accuracy first stays almost stable, then begins to degrade significantly when d is larger than 100km. This is because an increasing amount of far-away check-ins in the test set are considered to happen at hometown with the increasing d. From the observed changing trends, we can see that $d = 100$km is a threshold value since a distance around 100km is the typical radius of human "reach" – it takes about 1h to 2h to drive such a distance.

6.8. Recommendation Efficiency

This experiment is to evaluate the efficiency of our proposed online POI recommendation algorithm CBB on two real-life datasets. We compare it with two competitor algorithms and one baseline proposed in Section 6.2. All the online recommendation algorithms were implemented in Java (JDK 1.7) and run on a Windows Server 2008 with 256G RAM.

Table XI presents the average online efficiency of the four different methods over all queries created for $\mathcal{D}_{test}$ on the Twitter dataset. We show the performance with 1500 latent dimensions (i.e., $A = 1500$) and k set to 1, 5, 10, 15, and 20. A greater value of k is not necessary for the top-k recommendation task. Obviously, our proposed CBB algorithm outperforms others significantly and consistently for different numbers of recommendations. For example, on average, our proposed CBB algorithm finds the top-10 recommendations from about 62,000 POIs in 71.57ms, and achieves 1.56 times faster than the BF algorithm.

Table XII. Recommendation Efficiency on Foursquare Dataset

Methods	Online Recommendation Time Cost (ms)				
	k=1	k=5	k=10	k=15	k=20
CBB	**17.25**	**21.04**	**22.90**	**23.72**	**25.06**
BF	35.47	35.65	35.78	35.82	36.14
MT	41.89	42.07	42.20	42.36	42.56
TA	281.89	337.82	366.34	390.88	404.99

Table XIII. Impact of *B* on Twitter Dataset

	Online Recommendation Time Cost (ms)				
	k=1	k=5	k=10	k=15	k=20
B=10	71.16	86.79	94.47	97.85	103.38
B=50	62.54	76.27	83.02	85.99	90.85
B=100	54.77	66.8	72.72	75.32	79.57
B=200	53.91	65.75	71.57	74.13	78.32
B=500	53.89	65.72	71.54	74.1	78.29

Specifically, from the results, we observe that: (1) CBB outperforms BF significantly, justifying the benefits brought by pruning POI search space. It only needs to examine very few POIs (about 35% for $k = 10$, and 24% for $k = 5$). (2) Although the time cost of CBB increases with the increasing number of recommendations (i.e., k), it is still much lower than that of BF in the recommendation task even when $k = 20$. (3) The time cost of MT is higher than that of the naïve linear scan method (BF) in our task, although Koenigstein et al. [2012] reported the good performance of the MT in the setting with 50 attributes. This is because the MT loses its ability to prune POI search space and needs to scan all POIs in the leaf nodes when the dimensionality is high ($A > 200$). (4) The TA performs worse than the BF algorithm, because it needs to frequently update the threshold for each access of sorted lists and to maintain the dynamic priority queue of sorted lists, which is very time-consuming when the dimensionality of POIs is high since each attribute corresponds to a sorted list in the TA. These extra computations reduce down the efficiency of the TA when the dimensionality is high. In contrast, our CBB algorithm does not need to dynamically maintain a priority queue of sorted lists, nor frequently update a threshold score. In addition, our CBB needs to examine less POIs to find the accurate top-k results than the TA, since it groups POIs with similar directions into the same clusters. In summary, although both the TA and MT can achieve better performance than the BF due to their ability to prune POI search space when the dimensionality is not very high (e.g., less than 200), they cannot overcome the curse of dimensionality when the items have thousands of attributes, while our developed CBB algorithm can still achieve superior performance for the setting of high dimensionality.

Similar observations are also made on the Foursquare dataset, as shown in Table XII. The main difference is that the time costs of all the four algorithms are smaller than that on the Twitter dataset, because there are less available POIs in the Foursquare dataset and each POI has less attributes (i.e., $A = 800$ for the Foursquare dataset vs. $A = 1500$ for the Twitter dataset).

Since our CBB algorithm is sensitive to the number of buckets B, we evaluate its performance by varying B from 10 to 500 in this experiment. The experimental results are shown in Tables XIII and XIV. From the results, we observe that the time cost of CBB decreases with the increasing number of buckets B, and its performance does not change significantly when the number of buckets is larger than 100. At this point, it is less helpful to improve its performance by increasing B. Meanwhile, the clustering is

Table XIV. Impact of *B* on Foursquare Dataset

	Online Recommendation Time Cost (ms)				
	k=1	k=5	k=10	k=15	k=20
B=10	23.12	28.19	30.69	31.78	33.58
B=50	20.01	24.41	26.56	27.52	29.07
B=100	17.53	21.38	23.27	24.12	25.46
B=200	17.25	21.04	22.91	23.72	25.06
B=500	17.24	21.03	22.89	23.71	25.05

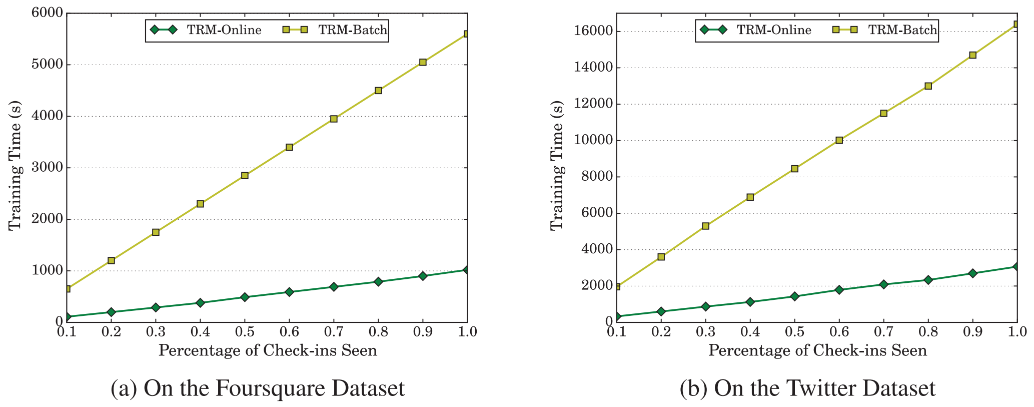


Fig. 7. Training time on the two datasets.

time-consuming when B becomes large. Thus, the performances reported in Tables XI and XII are achieved with $B = 200$. Note that, when B is too small or too big (e.g., $B = 1$ or $B = V$), our CBB algorithm would degenerate to the BF algorithm.

6.9. Model Training Efficiency

In this experiment, we evaluate the model training efficiency for both TRM-Batch and TRM-Online. We simulated the situation that check-ins are coming in a stream, and recorded the training time using currently observed check-ins at each decile point. When new check-ins arrive, TRM-Batch has to run over all check-ins for many iterations again. Since the time for each iteration grows with the number of check-in records, the total time for TRM-Batch grows fast. However, TRM-Online does not need to do this; it only needs to process the new coming check-ins and update parameters to get a new model. As Figure 7 shows, TRM-Batch costs much more time to get the new parameters compared with TRM-Online, especially when the number of observed check-ins is large. When running on the Foursquare dataset, the training time of TRM-Online is less than 1000s, while TRM-Batch takes more than 6,500s, which is about 6.5 times longer. On the Twitter dataset, TRM-Batch takes more than 16,000s, while TRM-Online takes less than 3,000s. From the results, we also observe that the training-time cost of our TRM-Online model linearly increases with the increasing number of check-ins; thus, it is scalable to large-scale datasets.

In summary, in addition to the benefit of the higher recommendation accuracy, as shown in Section 6.4, another advantage of TRM-Online is that it needs less training time.

7. RELATED WORK

This section introduces the related work, including POI recommendation, efficient model training, and efficient recommendation.

7.1. POI Recommendation

POI recommendation, also called place recommendation, has been considered as an essential task in the domain of recommender systems. It was first investigated and studied on trajectory data. Due to the lack of mapping relationship between geographical coordinates and specific real-world POIs, a POI is usually defined as the stay points extracted from users' trajectory logs [Zheng et al. 2009; Zheng and Xie 2011]. Because of the unavailability of content information associated with POIs, spatial and temporal patterns are commonly integrated into collaborative filtering methods to make POI recommendations. Recently, with the development of location-based social networks, it is easy for users to check in at POIs, resulting in easy access of large-scale user check-in records. Based on the LBSNs data, many recent works have tried to improve POI recommendation by exploiting and integrating geosocial, temporal, and semantic information associated with users' activities.

Geosocial Information. Many recent studies [Yin et al. 2016; Cheng et al. 2011; Ference et al. 2013; Ye et al. 2011; Zhu et al. 2015; Cho et al. 2011; Zhang et al. 2012] showed that there is a strong correlation between user check-in activities and geographical distance as well as social connections; thus, most of the current POI recommendation work mainly focuses on leveraging the geographical and social influences to improve recommendation accuracy. For example, Ye et al. [2011] delved into POI recommendation by investigating the geographical influences among locations and proposed a framework that combines user preferences, social influence, and geographical influence. Cheng et al. [2012] investigated the geographical influence through combining a multicenter Gaussian model, matrix factorization, and social influence for location recommendation. Lian et al. [2014] incorporated spatial clustering phenomena caused by geographical influence into a weighted matrix factorization framework to deal with the challenge of matrix sparsity. However, all of them do not consider the current location of the user. Thus, no matter whether users are located in the hometown or traveling out of town, they will recommend the same POIs to the users. In light of this, Ference et al. [2013] designed a collaborative recommendation framework that not only investigates the roles of friends in POI recommendation, but also considers the current location of the user.

Temporal Information. The temporal effect of user check-in activities in LBSNs has also attracted much attention from researchers. POI recommendations with temporal effect mainly leverage temporal cyclic patterns and temporal sequential patterns on LBSNs. Gao et al. [2013] investigated the temporal cyclic patterns of user check-ins in terms of temporal nonuniformness and temporal consecutiveness. Yuan et al. [2013] incorporated the temporal cyclic information into a user-based collaborative filtering framework for time-aware POI recommendation. Cheng et al. [2013] and Wang et al. [2016] focused on the task of successive personalized POI recommendation in LBSNs by embedding the temporal sequential patterns.

As described earlier, while there are many studies to improve POI recommendation by exploiting geographical-social influence and temporal effect, they did not address the challenges (e.g., data sparsity) arising from *user travel locality* for out-of-town recommendations. Most of this work assumed that users are in their hometowns; researchers did not consider users' real-time locations, nor their interest drift.

Semantic Information. Most recently, researchers explored the content information of POIs to alleviate the problem of data sparsity. Hu and Ester [2013] proposed a spatial topic model for POI recommendation considering both spatial aspect and textual aspect of user posts from Twitter. Liu and Xiong [2013] studied the effect of POI-associated tags for POI recommendation with an aggregated LDA and matrix factorization method. Yin et al. [2014, 2013] and Wang et al. [2015] exploited both personal interests and local preferences based on the contents associated with spatial items. Gao et al. [2015] and Zhao et al. [2015] studied both POI-associated contents and user sentiment information in relation to POI recommendation. However, all of them do not consider the time information associated with the contents of POIs.

As described earlier, while there are many studies to improve POI recommendation by exploiting the location, time, and content information to discover the patterns of users' activities in the spatial, temporal, and semantic aspects, respectively, they lack an integrated analysis of the joint effect of this heterogenous information to deal with data sparsity, especially in the out-of-town recommendation scenario, which has been ignored by most existing work. Our proposed TRM models strategically integrate all the heterogenous information to simultaneously discover the semantic, temporal, and spatial patterns in a unified fashion, which effectively overcomes the data sparsity and improves recommendation results by complementarity and mutual enhancement of the diverse information. Besides, TRM-Online is the first recommender model to support real-time POI recommendation by providing location-based, time-aware, and personalized services.

Although some recent literature [Baraglia et al. 2013; Noulas et al. 2012] used classification-based methods to predict the next place that a user will move by extracting multiple features from users' movement histories, their problem definition is different from ours. They assumed that the querying user is currently located at a POI, and exploited sequential pattern information to predict the next POI.

7.2. Efficient Model Training

Recently, real-time recommendation has attracted a lot of attention from both industry and academia [Chandramouli et al. 2011; Huang et al. 2015; Vinagre et al. 2014; Diaz-Aviles et al. 2012; Silva and Carin 2012; Lommatzsch and Albayrak 2015]. Traditional recommender algorithms focus on large user-item matrixes applying the collaborative filtering or matrix factorization models. In stream-based, real-time recommendation scenarios, these models cannot be applied due to tight time constraints and limited resources. To support real-time recommendations, various incremental online matrix-factorization or collaborative-filtering models are proposed [Chandramouli et al. 2011; Huang et al. 2015; Vinagre et al. 2014; Diaz-Aviles et al. 2012; Silva and Carin 2012; Lommatzsch and Albayrak 2015], and the efficiency of matrix-factorization models has also been extensively studied [Gemulla et al. 2011; Yu et al. 2012]. However, these conventional recommendation models based on collaborative filtering and matrix factorization do not perform well in the out-of-town recommendation setting, as analyzed in Yin et al. [2014] and Ference et al. [2013]. Meanwhile, the developed online model-training techniques for matrix factorization cannot be used to train our TRM model. This article focuses on the efficiency of LDA-like model training.

Most existing research on the LDA-like model utilizes various inference algorithms, such as variational Bayesian [Blei et al. 2003; Foulds et al. 2013], Gibbs sampling [Griffiths and Steyvers 2004], expectation propagation [Minka and Lafferty 2002], and belief propagation [Zeng et al. 2013] to obtain the parameters. Unfortunately, most are batch algorithms. Recently, a host of online learning methods have been developed for topic models. Some focus on modeling large amount of documents efficiently based on LDA. Typical research includes TM-LDA [Wang et al. 2012], On-Line LDA [AlSumait et al. 2008], and so on. By minimizing the error between predicted topic distribution and the real distribution generated by LDA, TM-LDA captures the latent topic transitions in temporal documents. On-Line LDA uses topics learned from previous documents by LDA as the prior of the following topics. It was designed by Alsumait et al. to detect and track topics in an online fashion. Some other works try to improve inference algorithms themselves. Banerjee and Basu [2007] presented online variants of vMF, EDCM, and LDA. Their experiments illustrated faster speed and better performance on real-world streaming text. Hoffman et al. [2010] developed an online variational Bayesian algorithm for LDA based on online stochastic optimization. In their approach, they thought of LDA as a probabilistic factorization of the matrix of word counts into a matrix of topic weights and a dictionary of topics. Thus, they used online matrix factorization techniques to obtain an online fashion schema of LDA. Canini et al. [2009] also proposed an online version algorithm using Gibbs sampling. Yao et al. [2009] compared several batch methods mentioned earlier, and introduced a new algorithm, SparseLDA, to accelerate the learning process.

In summary, a large number of prior works have made great efforts in designing appropriate online algorithms for LDA to process documents. However, fewer works focus on the check-in stream and tracking changing user interests and mobility patterns. In this article, we propose a novel online learning model TRM-Online to process check-in streams and to track changing user interests and mobility patterns for supporting real-time recommendation.

7.3. Efficient Recommendation

Tree Indexing-based Retrieval Algorithms. In most of the latent factor models for recommendation, such as matrix factorization and LDA-Like models, the ranking score of an item with regard to a query is computed as the inner product between their vectors. To the best of our knowledge, Koenigstein et al. [2012] were the first to pose and address the problem of fast maximum inner-product search. They proposed organizing the item vectors in an MT, in which each node is associated with a sphere that covers the item vectors below the node. Given a query vector, the spheres are exploited to avoid processing subtrees that cannot contribute to the result. The MT itself is constructed by repeatedly splitting the set of item vectors into two partitions (based on Euclidean distances). In the subsequent work [Curtin et al. 2013], the MT is replaced by a cover tree [Beygelzimer et al. 2006]. Both approaches effectively prune the item search space, but they suffer from high tree construction costs and from random memory access patterns during tree traversal. Moreover, according to the analysis in Liu et al. [2004], the speedup techniques using pure tree structures suffer from the curse of the dimensionality A, and their time cost is $\mathcal{O}(A^{12})$. Our experimental study also demonstrated that tree indexing-based algorithms are not effective for items with high dimensionality in practice.

TA-Based Algorithms. Yin et al. [2015b, 2014] extended the popular TA [Fagin et al. 2001] for top-k recommendation for monotonic functions. TA arranges the values of each coordinate of the item vectors in a sorted list, one per coordinate. Given a query, TA repeatedly selects a suitable list from a dynamically maintained priority queue, retrieves the next vector from the top of the list, and maintains the set of the top-k results seen so far. TA uses a termination criterion to stop processing as early as possible. Note that TA usually focuses on vectors of low dimensionality or medium size ($A < 200$), whereas we focus on vectors of high dimensionality ($A > 500$). TA needs to frequently update the threshold for each access of sorted lists and to maintain the dynamic priority queue of sorted lists. These extra computations reduce the efficiency of TA when the dimensionality is high. In contrast, our CBB algorithm need not dynamically maintain a priority queue of sorted lists, nor frequently update a threshold score. In addition, our CBB has a stronger pruning ability, as validated in the experiment.

Approximate Algorithms. Approximate methods for fast retrieval of top-k recommendations have also been studied in the literature. We categorize them as nonhashing methods and hashing methods. We first review nonhashing methods in Linden et al. [2003], who discussed the idea of item-space partitioning, which makes recommendations from some subsets of all items. They concluded that such a naïve strategy would produce recommendations of low quality. User clustering was adopted in Das et al. [2007] and Koenigstein et al. [2012] so that similar users share the same recommendation results. Although this strategy essentially reduces the user space, it also degrades the performances of personalized recommendations. Yin et al. [2014] extended TA to a ρ-approximation algorithm to speed up online recommendations and achieved a good trade-off between recommendation effectiveness and efficiency. Another line of work makes use of asymmetric transformations of user and item vectors to obtain an equivalent nearest-neighbor problem in Euclidean space; this problem is then solved approximately using LSH [Shrivastava and Li 2014] or modified PCA-trees [Bachrach et al. 2014]. An alternative approach is taken by Zhang et al. [2014], which modifies the classic matrix factorization model such that all vectors are (approximately) unit vectors and the inner product of user and item vectors can be approximated by standard cosine similarity search. However, this modification affects the quality of the recommendations.

8. CONCLUSIONS AND FUTURE WORK

In this article, we proposed a unified probabilistic generative *TRM* to simultaneously discover the semantic, temporal, and spatial patterns of users' check-in activities, and to model their joint effect on users' decision making for selection of POIs to visit. To demonstrate the applicability and flexibility of the TRM, we investigated how it supports two recommendation scenarios in a unified way, i.e., hometown recommendation and out-of-town recommendation. TRM can effectively overcomes the data sparsity and cold-start problems, especially when users travel out of town. To support real-time POI recommendation, we further extended the TRM model to an online learning model TRM-Online to capture the dynamics of user interests and accelerate the model training. Besides, based on the learnt model, we proposed a clustering-based branch and bound algorithm (CBB) to prune the POI search space and speed up the process of producing top-k recommendation. Compared with other existing pruning algorithms, the CBB algorithm has the desirable ability to overcome the curse of dimensionality.

To evaluate the performance of our proposals in the real-time POI recommendation, we conducted extensive experiments on two real-world datasets, including recommendation effectiveness, recommendation efficiency, and model-training efficiency. The experimental results demonstrated the superiority of our proposals, such as the TRM-Online model and the CBB retrieval algorithm, compared with the state-of-the-art competitors, by making more effective and efficient mobile recommendations. In addition, we studied the importance of each type of pattern in the two recommendation scenarios, respectively, and found that *semantic patterns* play a dominant role in overcoming the data sparsity in out-of-town recommendationw, while *temporal patterns* are most important to improve hometown recommendations.

The limitation of our proposed TMR model is that it assumes that users' interests are stable across geographical regions. But, in reality, users may have different interests when they travel to different regions (e.g., cities), especially when these regions have different urban compositions and cultures. In our future work, we will extend TRM to learn region-dependent personal interests according to the users' activity history at the specific region. However, due to travel locality, users' footprints left in each nonhome region are extremely sparse, which makes it very challenging to infer users' interests with regard to these regions. To alleviate the data sparsity of individual users at out-of-town regions, we will exploit the geographical and social correlations. Recent research [Cho et al. 2011] showed that there is a strong correlation between users' friendships and their mobility, especially when users travel out of town. Besides, if two regions are geographically proximate, the users' interests in these two regions should be similar, due to their similar urban compositions and cultures. Another promising research direction is to implement our online TRM model on the Spark platform to support large-scale and real-time POI recommendation.

APPENDIX

A. MODEL INFERENCE VIA COLLAPSED GIBBS SAMPLING

Here, we describe the inference algorithm for TRM based on the collapsed Gibbs sampling.

The task of posterior inference for the TRM is to determine the probability distributions of the hidden variables given the observed check-in records. However, exact inference is intractable due to the difficulty of calculating the normalizing constant in the joint distribution of the complete data, including both latent and observed variables, as shown in Equation (2). Therefore, we use collapsed Gibbs sampling, a well-established MCMC technique for approximate inference. In collapsed Gibbs sampling, the multinomial distributions $\boldsymbol{\Phi} = \{\boldsymbol{\theta}, \boldsymbol{\vartheta}, \boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{\varphi}\}$ are first marginalized (collapsed), and a Markov chain over the latent indicators $\{\boldsymbol{z}, \boldsymbol{r}\}$ is then constructed, whose stationary distribution is the posterior. We obtain samples of latent variables from the Markov chain. Point estimates for the collapsed distributions $\boldsymbol{\Phi}$ can then be computed given the samples, and predictive distributions are computed by averaging over multiple samples, as shown in Algorithm 2.

Sampling Procedure. The Gibbs sampler repeatedly samples latent variables conditioned on the current states of other hidden variables and observations; a configuration of latent states of the system is then obtained. Next, we provide the derivation of the sampling Equations (3) and (4).

Based on the generative process, we can get the posterior probability of the complete data by integrating out the multinomial distributions $\boldsymbol{\Phi} = \{\boldsymbol{\theta}, \boldsymbol{\vartheta}, \boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{\varphi}\}$, as shown in Equation (2), because the model uses only conjugate priors.

Sampling Topic. For the ith check-in record $(v^{ui}, l_v^{ui}, \mathcal{W}_v^{ui}, t^{ui})$ in $\mathcal{D}_u$, the conditional of z^{ui} is obtained by dividing the joint distribution of all variables but z^{ui} (denoted by $\boldsymbol{z}_{\neg}$) and canceling factors that do not depend on $\boldsymbol{z}_{\neg}$.

$$\begin{aligned} P(z^{ui}|\boldsymbol{z}_{\neg}, \boldsymbol{r}, \boldsymbol{v}, \boldsymbol{l}_v, \boldsymbol{\mathcal{W}}_v, \boldsymbol{t}, \cdot) &= \frac{P(\boldsymbol{v}, \boldsymbol{l}_v, \boldsymbol{\mathcal{W}}_v, \boldsymbol{t}, \boldsymbol{z}, \boldsymbol{r}|\cdot)}{P(\boldsymbol{v}, \boldsymbol{l}_v, \boldsymbol{\mathcal{W}}_v, \boldsymbol{t}, \boldsymbol{z}_{\neg}, \boldsymbol{r}|\cdot)} \\ &= \frac{P(\boldsymbol{z}|\alpha)}{P(\boldsymbol{z}_{\neg}|\alpha)} \cdot \frac{P(\boldsymbol{\mathcal{W}}_v|\boldsymbol{z}, \beta)}{P(\boldsymbol{\mathcal{W}}_v|\boldsymbol{z}_{\neg}, \beta)} \cdot \frac{P(\boldsymbol{t}|\boldsymbol{z}, \eta)}{P(\boldsymbol{t}|\boldsymbol{z}_{\neg}, \eta)} \cdot \frac{P(\boldsymbol{v}|\boldsymbol{z}, \boldsymbol{r}, \tau)}{P(\boldsymbol{v}|\boldsymbol{z}_{\neg}, \boldsymbol{r}, \tau)} \\ &= \frac{P(\boldsymbol{z}|\alpha)}{P(\boldsymbol{z}_{\neg}|\alpha)} \cdot \prod_{\boldsymbol{w} \in \boldsymbol{\mathcal{W}}_v} \frac{P(\boldsymbol{w}|\boldsymbol{z}, \beta)}{P(\boldsymbol{w}|\boldsymbol{z}_{\neg}, \beta)} \cdot \frac{P(\boldsymbol{t}|\boldsymbol{z}, \eta)}{P(\boldsymbol{t}|\boldsymbol{z}_{\neg}, \eta)} \cdot \frac{P(\boldsymbol{v}|\boldsymbol{z}, \boldsymbol{r}, \tau)}{P(\boldsymbol{v}|\boldsymbol{z}_{\neg}, \boldsymbol{r}, \tau)} \end{aligned} \quad (32)$$

We now derive the first fraction of Equation (32), that is,

$$\frac{P(\boldsymbol{z}|\alpha)}{P(\boldsymbol{z}_{\neg}|\alpha)} = \frac{\int P(\boldsymbol{z}|\boldsymbol{\theta})P(\boldsymbol{\theta}|\alpha)d\boldsymbol{\theta}}{\int P(\boldsymbol{z}_{\neg}|\boldsymbol{\theta})P(\boldsymbol{\theta}|\alpha)d\boldsymbol{\theta}}. \quad (33)$$

As we assume that each topic z is generated from a multinomial distribution $\boldsymbol{\theta}$, and the hyper-parameter for conjugate Dirichlet prior is α, we have that

$$\begin{aligned}
&\int P(\boldsymbol{z}|\boldsymbol{\theta})P(\boldsymbol{\theta}|\alpha)d\boldsymbol{\theta} \\
&= \int \prod_{u\in\mathcal{U}}\prod_{i=1}^{D_u} P(z^{ui}|\boldsymbol{\theta}_u)\prod_{u\in\mathcal{U}} P(\boldsymbol{\theta}_u|\alpha)d\boldsymbol{\theta} \\
&= \int \prod_{u\in\mathcal{U}} \frac{\Gamma(\sum_z \alpha)}{\prod_z \Gamma(\alpha)}\prod_z \theta_{u,z}^{n_{u,z}+\alpha-1}d\boldsymbol{\theta} \\
&= \prod_{u\in\mathcal{U}} \frac{\Gamma(\sum_z \alpha)}{\prod_z \Gamma(\alpha)} \cdot \frac{\prod\limits_z \Gamma(n_{u,z}+\alpha)}{\Gamma(\sum\limits_z (n_{u,z}+\alpha))}.
\end{aligned}$$

Combining this equation with Equation (33) leads to

$$\begin{aligned}
\frac{P(\boldsymbol{z}|\alpha)}{P(\boldsymbol{z}_{\neg}|\alpha)} &= \frac{\Gamma(n_{u,z^{ui}}+\alpha)\Gamma(\sum\limits_z (n_{u,z,\neg}+\alpha))}{\Gamma(\sum\limits_z (n_{u,z}+\alpha))\Gamma(n_{u,z^{ui},\neg}+\alpha)}, \\
&= \frac{n_{u,z^{ui},\neg}+\alpha}{\sum_z (n_{u,z,\neg}+\alpha)}
\end{aligned} \tag{34}$$

where the count with subscript $\neg$ denotes a quantity excluding the current instance (i.e., the ith check-in record in $\mathcal{D}_u$). Here, we use the identity $\Gamma(x+1) = x\Gamma(x)$. The second, third, and fourth fractions of Equation (32) can be derived analogously. The Dirichlet-Multinomial conjugates ensure the tractability of the integrals. Specifically, the second fraction can be written as

$$\begin{aligned}
\frac{P(\mathcal{W}_v|\boldsymbol{z},\beta)}{P(\mathcal{W}_v|\boldsymbol{z}_{\neg},\beta)} = \prod_{\boldsymbol{w}\in\mathcal{W}_v} \frac{P(\boldsymbol{w}|\boldsymbol{z},\beta)}{P(\boldsymbol{w}|\boldsymbol{z}_{\neg},\beta)} &= \prod_{\boldsymbol{w}\in\mathcal{W}_v} \frac{\int P(\boldsymbol{w}|\boldsymbol{z},\boldsymbol{\phi})P(\boldsymbol{\phi}|\beta)d\boldsymbol{\phi}}{\int P(\boldsymbol{w}|\boldsymbol{z}_{\neg},\boldsymbol{\phi})P(\boldsymbol{\phi}|\beta)d\boldsymbol{\phi}} \\
&= \prod_{w^{ui}\in\mathcal{W}_v^{ui}} \frac{\Gamma(n_{z^{ui},w^{ui}}+\beta)\Gamma(\sum\limits_w (n_{z^{ui},w,\neg}+\beta))}{\Gamma(\sum\limits_w (n_{z^{ui},w}+\beta))\Gamma(n_{z^{ui},w^{ui},\neg}+\beta)} \\
&= \prod_{w^{ui}\in\mathcal{W}_v^{ui}} \frac{n_{z^{ui},w^{ui},\neg}+\beta}{\sum_w (n_{z^{ui},w,\neg}+\beta)}.
\end{aligned} \tag{35}$$

The third fraction can be written as

$$\begin{aligned}
\frac{P(\boldsymbol{t}|\boldsymbol{z},\eta)}{P(\boldsymbol{t}|\boldsymbol{z}_{\neg},\eta)} &= \frac{\int P(\boldsymbol{t}|\boldsymbol{z},\boldsymbol{\psi})P(\boldsymbol{\psi}|\eta)d\boldsymbol{\psi}}{\int P(\boldsymbol{t}|\boldsymbol{z}_{\neg},\boldsymbol{\psi})P(\boldsymbol{\psi}|\eta)d\boldsymbol{\psi}} \\
&= \frac{\Gamma(n_{z^{ui},t^{ui}}+\eta)\Gamma(\sum\limits_t (n_{z^{ui},t,\neg}+\eta))}{\Gamma(\sum\limits_t (n_{z^{ui},t}+\eta))\Gamma(n_{z^{ui},t^{ui},\neg}+\eta)} \\
&= \frac{n_{z^{ui},t^{ui},\neg}+\eta}{\sum_t (n_{z^{ui},t,\neg}+\eta)}.
\end{aligned} \tag{36}$$

The fourth fraction can be written as

$$\begin{aligned}\frac{P(\boldsymbol{v}|\boldsymbol{z},\boldsymbol{r},\tau)}{P(\boldsymbol{v}|\boldsymbol{z}_\neg,\boldsymbol{r},\tau)} &= \frac{\int P(\boldsymbol{v}|\boldsymbol{z},\boldsymbol{r},\boldsymbol{\varphi})P(\boldsymbol{\varphi}|\tau)d\boldsymbol{\varphi}}{\int P(\boldsymbol{v}|\boldsymbol{z}_\neg,\boldsymbol{r},\boldsymbol{\varphi})P(\boldsymbol{\varphi}|\tau)d\boldsymbol{\varphi}}\\ &= \frac{\Gamma(n_{z^{ui},r^{ui},v^{ui}}+\tau)\Gamma(\sum\limits_{v}(n_{z^{ui},r^{ui},v,\neg}+\tau))}{\Gamma(\sum\limits_{v}(n_{z^{ui},r^{ui},v}+\tau))\Gamma(n_{z^{ui},r^{ui},v^{ui},\neg}+\tau)}\\ &= \frac{n_{z^{ui},r^{ui},v^{ui},\neg}+\tau}{\sum_v(n_{z^{ui},r^{ui},v,\neg}+\tau)}.\end{aligned} \tag{37}$$

Finally, by combining Equations (32) to (41), we obtain the sampling formula, as follows:

$$\begin{aligned}&P(z^{ui}|\boldsymbol{z}_\neg,\boldsymbol{r},\boldsymbol{v},\boldsymbol{l_v},\boldsymbol{\mathcal{W}_v},\boldsymbol{t},\cdot)\\ &\propto \frac{n_{u,z^{ui},\neg}+\alpha}{\sum_z(n_{u,z,\neg}+\alpha)}\prod_{w^{ui}\in\mathcal{W}_v^{ui}}\frac{n_{z^{ui},w^{ui},\neg}+\beta}{\sum_w(n_{z^{ui},w,\neg}+\beta)}\frac{n_{z^{ui},t^{ui},\neg}+\eta}{\sum_t(n_{z^{ui},t,\neg}+\eta)}\frac{n_{z^{ui},r^{ui},v^{ui},\neg}+\tau}{\sum_v(n_{z^{ui},r^{ui},v,\neg}+\tau)}.\end{aligned}$$

For clear presentation, this formula can be rewritten as in Equation (3).

Sampling Region. For the ith check-in record $(v^{ui}, l_v^{ui}, \mathcal{W}_v^{ui}, t^{ui})$ in $\mathcal{D}_u$, the conditional of $r^{u,i}$ is obtained by dividing the joint distribution of all variables but r^{ui} (denoted by $\boldsymbol{r}_\neg$) and canceling factors that do not depend on $\boldsymbol{r}_\neg$.

$$\begin{aligned}&P(r^{ui}|\boldsymbol{z},\boldsymbol{r}_\neg,\boldsymbol{v},\boldsymbol{l_v},\boldsymbol{\mathcal{W}_v},\boldsymbol{t},\cdot)\\ &= \frac{P(\boldsymbol{v},\boldsymbol{l_v},\boldsymbol{\mathcal{W}_v},\boldsymbol{t},\boldsymbol{z},\boldsymbol{r}|\cdot)}{P(\boldsymbol{v},\boldsymbol{l_v},\boldsymbol{\mathcal{W}_v},\boldsymbol{t},\boldsymbol{z},\boldsymbol{r}_\neg|\cdot)}\\ &= \frac{P(\boldsymbol{r}|\gamma)}{P(\boldsymbol{r}_\neg|\gamma)}\cdot\frac{P(\boldsymbol{v}|\boldsymbol{z},\boldsymbol{r},\tau)}{P(\boldsymbol{v}|\boldsymbol{z},\boldsymbol{r}_\neg,\tau)}\cdot\frac{P(\boldsymbol{l_v}|\boldsymbol{r},\boldsymbol{\mu},\boldsymbol{\Sigma})}{P(\boldsymbol{l_v}|\boldsymbol{r}_\neg,\boldsymbol{\mu},\boldsymbol{\Sigma})}.\end{aligned} \tag{38}$$

We now derive the first fraction of Equation (38), that is,

$$\frac{P(\boldsymbol{r}|\gamma)}{P(\boldsymbol{r}_\neg|\gamma)} = \frac{\int P(\boldsymbol{r}|\boldsymbol{\vartheta})P(\boldsymbol{\vartheta}|\gamma)d\boldsymbol{\vartheta}}{\int P(\boldsymbol{r}_\neg|\boldsymbol{\vartheta})P(\boldsymbol{\vartheta}|\gamma)d\boldsymbol{\vartheta}}. \tag{39}$$

As we assume that each region r is generated from a multinomial distribution $\boldsymbol{\vartheta}$, and the hyper-parameter for conjugate Dirichlet prior is γ, we have that

$$\begin{aligned}&\int P(\boldsymbol{r}|\boldsymbol{\vartheta})P(\boldsymbol{\vartheta}|\gamma)d\boldsymbol{\vartheta}\\ &= \int\prod_{u\in\mathcal{U}}\prod_{i=1}^{D_u}P\big(r^{ui}|\boldsymbol{\vartheta}_u\big)\prod_{u\in\mathcal{U}}P(\boldsymbol{\vartheta}_u|\gamma)d\boldsymbol{\vartheta}\\ &= \int\prod_{u\in\mathcal{U}}\frac{\Gamma(\sum_r\gamma)}{\prod_r\Gamma(\gamma)}\prod_r\vartheta_{u,r}^{n_{u,r}+\gamma-1}d\boldsymbol{\vartheta}\\ &= \prod_{u\in\mathcal{U}}\frac{\Gamma(\sum_r\gamma)}{\prod_r\Gamma(\gamma)}\cdot\frac{\prod\limits_r\Gamma(n_{u,r}+\gamma)}{\Gamma(\sum\limits_r(n_{u,r}+\gamma))}.\end{aligned}$$

Combining this equation with Equation (39) leads to

$$\begin{aligned}\frac{P(\boldsymbol{r}|\gamma)}{P(\boldsymbol{r}_{\neg}|\gamma)} &= \frac{\Gamma(n_{u,r^{ui}}+\gamma)\Gamma(\sum_{r}(n_{u,r,\neg}+\gamma))}{\Gamma(\sum_{r}(n_{u,r}+\gamma))\Gamma(n_{u,r^{ui},\neg}+\gamma)} \\ &= \frac{n_{u,r^{ui},\neg}+\gamma}{\sum_r(n_{u,r,\neg}+\gamma)}.\end{aligned} \tag{40}$$

The second fraction of Equation (38) can be derived analogously. The Dirichlet-Multinomial conjugates ensure the tractability of the integrals. Specifically, the second fraction can be written as

$$\begin{aligned}\frac{P(\boldsymbol{v}|\boldsymbol{z},\boldsymbol{r},\tau)}{P(\boldsymbol{v}|\boldsymbol{z},\boldsymbol{r}_{\neg},\tau)} &= \frac{\int P(\boldsymbol{v}|\boldsymbol{z},\boldsymbol{r},\boldsymbol{\varphi})P(\boldsymbol{\varphi}|\tau)d\boldsymbol{\varphi}}{\int P(\boldsymbol{v}|\boldsymbol{z},\boldsymbol{r}_{\neg},\boldsymbol{\varphi})P(\boldsymbol{\varphi}|\tau)d\boldsymbol{\varphi}} \\ &= \frac{\Gamma(n_{z^{ui},r^{ui},v^{ui}}+\tau)\Gamma(\sum_{v}(n_{z^{ui},r^{ui},v,\neg}+\tau))}{\Gamma(\sum_{v}(n_{z^{ui},r^{ui},v}+\tau))\Gamma(n_{z^{ui},r^{ui},v^{ui},\neg}+\tau)} \\ &= \frac{n_{z^{ui},r^{ui},v^{ui},\neg}+\tau}{\sum_v(n_{z^{ui},r^{ui},v,\neg}+\tau)}.\end{aligned} \tag{41}$$

As we assume that each geographical coordinate l_v is generated by a region-specific Gaussian distribution, we have that

$$P(\boldsymbol{l_v}|\boldsymbol{r},\boldsymbol{\mu},\boldsymbol{\Sigma}) = \prod_{u\in\mathcal{U}}\prod_{i=1}^{D_u}\mathcal{N}(\boldsymbol{\mu}_{r_{u,i}},\boldsymbol{\Sigma}_{r_{u,i}}).$$

Based on this equation, the third fraction of Equation (38) can be written as

$$\frac{P(\boldsymbol{l_v}|\boldsymbol{r},\boldsymbol{\mu},\boldsymbol{\Sigma})}{P\big(\boldsymbol{l_v}|\boldsymbol{r}_{\neg},\boldsymbol{\mu},\boldsymbol{\Sigma}\big)} = P(l_v^{ui}|\boldsymbol{\mu}_{r^{ui}},\boldsymbol{\Sigma}_{r^{ui}}). \tag{42}$$

Finally, by combining Equations (38) to (42), we obtain the sampling formula, as follows:

$$\begin{aligned}&P(r^{ui}|\boldsymbol{z},\boldsymbol{r}_{\neg},\boldsymbol{v},\boldsymbol{l_v},\boldsymbol{\mathcal{W}_v},\boldsymbol{t},\cdot) \\ &\quad\propto \frac{n_{u,r^{ui},\neg}+\gamma}{\sum_r(n_{u,r,\neg}+\gamma)}\frac{n_{z^{ui},r^{ui},v^{ui},\neg}+\tau}{\sum_v(n_{z^{ui},r^{ui},v,\neg}+\tau)}P\big(l_v^{ui}|\boldsymbol{\mu}_{r^{ui}},\boldsymbol{\Sigma}_{r^{ui}}\big).\end{aligned}$$

For clarity, this sampling formula can be rewritten as in Equation (4).

REFERENCES

Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering* 17, 6, 734–749.

Amr Ahmed, Yucheng Low, Mohamed Aly, Vanja Josifovski, and Alexander J. Smola. 2011. Scalable distributed inference of dynamic user interests for behavioral targeting. In *KDD*. 114–122.

Loulwah AlSumait, Daniel Barbar, and Carlotta Domeniconi. 2008. On-line LDA: Adaptive topic models for mining text streams with applications to topic detection and tracking. In *ICDM*. 3–12.

Yoram Bachrach, Yehuda Finkelstein, Ran Gilad-Bachrach, Liran Katzir, Noam Koenigstein, Nir Nice, and Ulrich Paquet. 2014. Speeding up the Xbox recommender system using a Euclidean transformation for inner-product spaces. In *RecSys*. 257–264.

Arindam Banerjee and Sugato Basu. 2007. Topic models over text streams: A study of batch and online unsupervised learning. In *SDM*, Vol. 7. 437–442.

Jie Bao, Yu Zheng, and Mohamed F. Mokbel. 2012. Location-based and preference-aware recommendation using sparse geo-social networking data. In *SIGSPATIAL*. 199–208.

Ranieri Baraglia, Cristina Ioana Muntean, Franco Maria Nardini, and Fabrizio Silvestri. 2013. LearNext: Learning to predict tourists movements. In *CIKM*. 751–756.

Alina Beygelzimer, Sham Kakade, and John Langford. 2006. Cover trees for nearest neighbor. In *ICML*. 97–104.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning and Research* 3, 993–1022.

Pedro G. Campos, Fernando Díez, and Iván Cantador. 2014. Time-aware recommender systems: A comprehensive survey and analysis of existing evaluation protocols. *User Modeling and User-Adapted Interaction* 24, 1–2, 67–119.

Kevin R. Canini, Lei Shi, and Thomas L. Griffiths. 2009. Online inference of topics with latent Dirichlet allocation. In *AISTATS*. 65–72.

Badrish Chandramouli, Justin J. Levandoski, Ahmed Eldawy, and Mohamed F. Mokbel. 2011. StreamRec: A real-time recommender system. In *SIGMOD*. 1243–1246.

Tianqi Chen, Weinan Zhang, Qiuxia Lu, Kailong Chen, Zhao Zheng, and Yong Yu. 2012. SVDFeature: A toolkit for feature-based collaborative filtering. *Journal of Machine Learning and Research* 13, 1, 3619–3622.

Chen Cheng, Haiqin Yang, Irwin King, and Michael R. Lyu. 2012. Fused matrix factorization with geographical and social influence in location-based social networks. In *AAAI*.

Chen Cheng, Haiqin Yang, Michael R. Lyu, and Irwin King. 2013. Where you like to go next: Successive point-of-interest recommendation. In *IJCAI*. 2605–2611.

Zhiyuan Cheng, James Caverlee, Kyumin Lee, and Daniel Z. Sui. 2011. Exploring millions of footprints in location sharing services. In *ICWSM*.

Eunjoon Cho, Seth A. Myers, and Jure Leskovec. 2011. Friendship and mobility: User movement in location-based social networks. In *KDD*. 1082–1090.

Ryan R. Curtin, Alexander G. Gray, Parikshit Ram, Ryan R. Curtin, Parikshit Ram Alexander, and G. Gray. 2013. Fast exact max-kernel search. In *SDM*. 1–9.

Abhinandan S. Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. 2007. Google news personalization: Scalable online collaborative filtering. In *WWW*. 271–280.

Inderjit S. Dhillon and Dharmendra S. Modha. 2001. Concept decompositions for large sparse text data using clustering. *Machine Learning* 42, 1–2, 143–175.

Ernesto Diaz-Aviles, Lucas Drumond, Lars Schmidt-Thieme, and Wolfgang Nejdl. 2012. Real-time top-n recommendation in social streams. In *RecSys*. 59–66.

Arnaud Doucet, Nando de Freitas, Kevin P. Murphy, and Stuart J. Russell. 2000a. Rao-blackwellised particle filtering for dynamic Bayesian networks. In *UAI*. 176–183.

Arnaud Doucet, Simon Godsill, and Christophe Andrieu. 2000b. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing* 10, 3, 197–208.

Ronald Fagin, Amnon Lotem, and Moni Naor. 2001. Optimal aggregation algorithms for middleware. In *PODS*. 102–113.

Gregory Ference, Mao Ye, and Wang-Chien Lee. 2013. Location recommendation for out-of-town users in location-based social networks. In *CIKM*. 721–726.

James Foulds, Levi Boyles, Christopher DuBois, Padhraic Smyth, and Max Welling. 2013. Stochastic collapsed variational Bayesian inference for latent Dirichlet allocation. In *KDD*. 446–454.

Huiji Gao, Jiliang Tang, Xia Hu, and Huan Liu. 2013. Exploring temporal effects for location recommendation on location-based social networks. In *RecSys*. 93–100.

Huiji Gao, Jiliang Tang, Xia Hu, and Huan Liu. 2015. Content-aware point of interest recommendation on location-based social networks. In *AAAI*.

Huiji Gao, Jiliang Tang, and Huan Liu. 2012. gSCorr: Modeling geo-social correlations for new check-ins on location-based social networks. In *CIKM*. 1582–1586.

Rainer Gemulla, Erik Nijkamp, Peter J. Haas, and Yannis Sismanis. 2011. Large-scale matrix factorization with distributed stochastic gradient descent. In *KDD*. 69–77.

Thomas L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences* 101, suppl. 1, 5228–5235.

Matthew Hoffman, Francis R. Bach, and David M. Blei. 2010. Online learning for latent Dirichlet allocation. In *NIPS*. 856–864.

Liangjie Hong, Amr Ahmed, Siva Gurumurthy, Alexander J. Smola, and Kostas Tsioutsiouliklis. 2012. Discovering geographical topics in the Twitter stream. In *WWW*. 769–778.

Bo Hu and Martin Ester. 2013. Spatial topic modeling in online social media for location recommendation. In *RecSys*. 25–32.

Yanxiang Huang, Bin Cui, Wenyu Zhang, Jie Jiang, and Ying Xu. 2015. TencentRec: Real-time stream recommendation in practice. In *SIGMOD*. 227–238.

Noam Koenigstein, Parikshit Ram, and Yuval Shavitt. 2012. Efficient retrieval of recommendations in a matrix factorization framework. In *CIKM*. 535–544.

Justin J. Levandoski, Mohamed Sarwat, Ahmed Eldawy, and Mohamed F. Mokbel. 2012. LARS: A location-aware recommender system. In *ICDE*. 450–461.

Rui Li, Shengjie Wang, Hongbo Deng, Rui Wang, and Kevin Chen-Chuan Chang. 2012. Towards social user profiling: Unified and discriminative influence model for inferring home locations. In *KDD*. 1023–1031.

Defu Lian, Cong Zhao, Xing Xie, Guangzhong Sun, Enhong Chen, and Yong Rui. 2014. GeoMF: Joint geographical modeling and matrix factorization for point-of-interest recommendation. In *KDD*. 831–840.

Moshe Lichman and Padhraic Smyth. 2014. Modeling human location data with mixtures of kernel densities. In *KDD*. 35–44.

Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing* 7, 1, 76–80.

Bin Liu, Yanjie Fu, Zijun Yao, and Hui Xiong. 2013. Learning geographical preferences for point-of-interest recommendation. In *KDD*. 1043–1051.

Bin Liu and Hui Xiong. 2013. Point-of-interest recommendation in location based social networks with topic and location awareness. In *SDM*. 396–404.

Ting Liu, Andrew W. Moore, Ke Yang, and Alexander G. Gray. 2004. An investigation of practical approximate nearest neighbor algorithms. In *NIPS*. 825–832.

Andreas Lommatzsch and Sahin Albayrak. 2015. Real-time recommendations for user-item streams. In *SAC*. 1039–1046.

Thomas Minka and John Lafferty. 2002. Expectation-propagation for the generative aspect model. In *UAI*. 352–359.

Diana Mok, Barry Wellman, and Juan Carrasco. 2010. Does distance matter in the age of the Internet? *Urban Studies* 47, 13, 2747–2783.

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

Salvatore Scellato, Anastasios Noulas, Renaud Lambiotte, and Cecilia Mascolo. 2011. Socio-spatial properties of online location-based social networks. In *ICWSM*.

Anshumali Shrivastava and Ping Li. 2014. Asymmetric lsh (alsh) for sublinear time maximum inner product search (MIPS). In *NIPS*. 2321–2329.

Jorge Silva and Lawrence Carin. 2012. Active learning for online Bayesian matrix factorization. In *KDD*. 325–333.

Alexander Smola and Shravan Narayanamurthy. 2010. An architecture for parallel topic models. *Proceedings of the VLDB Endowment* 3, 1–2, 703–710.

Jie Tang, Sen Wu, Jimeng Sun, and Hang Su. 2012. Cross-domain collaboration recommendation. In *KDD*. 1285–1293.

Christina Teflioudi, Rainer Gemulla, and Olga Mykytiuk. 2015. LEMP: Fast retrieval of large entries in a matrix product. In *SIGMOD*. 107–122.

João Vinagre, Alípio Mário Jorge, and João Gama. 2014. Fast incremental matrix factorization for recommendation with positive-only feedback. In *User Modeling, Adaptation, and Personalization*. 459–470.

Hanna M. Wallach, David M. Mimno, and Andrew McCallum. 2009. Rethinking LDA: Why priors matter. In *NIPS*. 1973–1981.

Weiqing Wang, Hongzhi Yin, Ling Chen, Yizhou Sun, Shazia Sadiq, and Xiaofang Zhou. 2015. Geo-SAGE: A geographical sparse additive generative model for spatial item recommendation. In *KDD*. 1255–1264.

Weiqing Wang, Hongzhi Yin, Shazia Sadiq, Ling Chen, Min Xie, and Xiaofang Zhou. 2016. SPORE: A sequential personalized spatial item recommender system. In *ICDE*. 12.

Yu Wang, Eugene Agichtein, and Michele Benzi. 2012. TM-LDA: Efficient online modeling of latent topic transitions in social media. In *KDD*. 123–131.

Limin Yao, David Mimno, and Andrew McCallum. 2009. Efficient methods for topic model inference on streaming document collections. In *KDD*. 937–946.

Mao Ye, Dong Shou, Wang-Chien Lee, Peifeng Yin, and Krzysztof Janowicz. 2011. On the semantic annotation of places in location-based social networks. In *KDD*. 520–528.

Mao Ye, Peifeng Yin, and Wang-Chien Lee. 2010. Location recommendation for location-based social networks. In *GIS*. 458–461.

Mao Ye, Peifeng Yin, Wang-Chien Lee, and Dik-Lun Lee. 2011. Exploiting geographical influence for collaborative point-of-interest recommendation. In *SIGIR*. 325–334.

Hongzhi Yin, Bin Cui, Ling Chen, Zhiting Hu, and Chengqi Zhang. 2015a. Modeling location-based user rating profiles for personalized recommendation. *ACM Transactions on Knowledge Discovery from Data* 9, 3, Article 19, 41 pages.

Hongzhi Yin, Bin Cui, Ling Chen, Zhiting Hu, and Xiaofang Zhou. 2015b. Dynamic user modeling in social media systems. *ACM Transactions on Information Systems* 33, 3, Article 10, 44 pages.

Hongzhi Yin, Bin Cui, Zi Huang, Weiqing Wang, Xian Wu, and Xiaofang Zhou. 2015c. Joint modeling of users' interests and mobility patterns for point-of-interest recommendation. In *MM*. 819–822.

Hongzhi Yin, Bin Cui, Yizhou Sun, Zhiting Hu, and Ling Chen. 2014. LCARS: A spatial item recommender system. *ACM Transactions on Information Systems* 32, 3, Article 11, 37 pages.

Hongzhi Yin, Zhiting Hu, Xiaofang Zhou, Hao Wang, Kai Zheng, Hung Nguyen Quoc Viet, and Shazia Sadiq. 2016. Discovering interpretable geo-social communities for user behavior prediction. In *ICDE*. 12.

Hongzhi Yin, Yizhou Sun, Bin Cui, Zhiting Hu, and Ling Chen. 2013. LCARS: A location-content-aware recommender system. In *KDD*. 221–229.

Hongzhi Yin, Xiaofang Zhou, Yingxia Shao, Hao Wang, and Shazia Sadiq. 2015. Joint modeling of user check-in behaviors for point-of-interest recommendation. In *CIKM*. 1631–1640.

Zhijun Yin, Liangliang Cao, Jiawei Han, Chengxiang Zhai, and Thomas Huang. 2011. Geographical topic discovery and comparison. In *WWW*. 247–256.

Hsiang-Fu Yu, Cho-Jui Hsieh, Si Si, and Inderjit S. Dhillon. 2012. Scalable coordinate descent approaches to parallel matrix factorization for recommender systems. In *ICDM*. 765–774.

Quan Yuan, Gao Cong, Zongyang Ma, Aixin Sun, and Nadia Magnenat Thalmann. 2013. Time-aware Point-of-interest Recommendation. In *SIGIR*. 363–372.

Jia Zeng, W. K. Cheung, and Jiming Liu. 2013. Learning topic models by belief propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 5, 1121–1134.

Chao Zhang, Lidan Shou, Ke Chen, Gang Chen, and Yijun Bei. 2012. Evaluating geo-social influence in location-based social networks. In *CIKM*. 1442–1451.

Zhiwei Zhang, Qifan Wang, Lingyun Ruan, and Luo Si. 2014. Preference preserving hashing for efficient recommendation. In *SIGIR*. 183–192.

Kaiqi Zhao, Gao Cong, Quan Yuan, and Kenny Zhu. 2015. SAR: A sentiment-aspect-region model for user preference analysis in geo-tagged reviews. In *ICDE*. 12.

Wayne Xin Zhao, Jing Jiang, Jianshu Weng, Jing He, Ee-Peng Lim, Hongfei Yan, and Xiaoming Li. 2011. Comparing Twitter and traditional media using topic models. In *ECIR*. 338–349.

Yu Zheng and Xing Xie. 2011. Learning travel recommendations from user-generated GPS traces. *ACM Transactions on Intelligent Systems and Technology* 2, 1, 2:1–2:29.

Yu Zheng, Lizhu Zhang, Xing Xie, and Wei-Ying Ma. 2009. Mining interesting locations and travel sequences from GPS trajectories. In *WWW*. 791–800.

Ke Zhou and Hongyuan Zha. 2012. Learning binary codes for collaborative filtering. In *KDD*. 498–506.

Wen-Yuan Zhu, Wen-Chih Peng, Ling-Jyh Chen, Kai Zheng, and Xiaofang Zhou. 2015. Modeling user mobility for location promotion in location-based social networks. In *KDD*. 1573–1582.

Received August 2015; revised November 2015; accepted December 2015