SPRINGER BRIEFS IN COMPUTER SCIENCE

Hongzhi Yin
Bin Cui

Spatio-Temporal Recommendation in Social Media

Springer

SpringerBriefs in Computer Science

More information about this series at http://www.springer.com/series/10028

Hongzhi Yin · Bin Cui

Spatio-Temporal Recommendation in Social Media

Springer

Hongzhi Yin
The University of Queensland
Brisbane, QLD
Australia

Bin Cui
Peking University
Beijing
China

ISSN 2191-5768 ISSN 2191-5776 (electronic)
SpringerBriefs in Computer Science
ISBN 978-981-10-0747-7 ISBN 978-981-10-0748-4 (eBook)
DOI 10.1007/978-981-10-0748-4

Library of Congress Control Number: 2016939068

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer Science+Business Media Singapore Pte Ltd.

To all who make our lives worthwhile

Preface

In this book, we will guide you through the world of spatiotemporal recommendation in social media, which aims to help users find their potentially preferred items by mining the spatiotemporal data generated by the users in social media sites and apps. The spatiotemporal data imply extensive knowledge about individuals' behaviors, mobility, and interests. It also bridges the gap between the online social networks and the physical world, which enables us to better understand the users, improve user experiences, and design optimal recommendation systems. Targeted advertisement recommendation in social media is one of the application scenarios, which is predicted to generate hundreds of billions of dollars revenue.

However, spatiotemporal recommendation in social media is a highly challenging research problem because of the temporal dynamics of users' behaviors and interests, users' interests drift over geographical regions, data sparsity and cold start in the specific spatiotemporal contexts (e.g., when users travel out of town or to new cities). Moreover, users' generated spatiotemporal data in social media arrives in a timely fashion (e.g., data stream), making this problem much more difficult. Most traditional recommender techniques encounter various limitations and insufficiency.

Our book covers the major fundamentals and the state-of-the-art research of new generation spatiotemporal recommendation system in social media. This book provides researchers and developers a rich blend of theory and practice to help them explore this exciting field and develop new methods and application scenarios. It is also suitable for advanced undergraduates and graduate students, since each chapter is a tutorial that provides readers with an introduction to one important aspect of spatiotemporal recommendation in social media and also contains many valuable references to relevant research papers.

February 2016

Hongzhi Yin
Bin Cui

Acknowledgments

This work is partially supported by the Discovery Early Career Researcher Award (DE160100308) and ARC Discovery Project (DP140103171). It is also partially supported by National Natural Science Foundation of China (Grant No. 61572335, 61272155, 61232006, 61303164, 61402447) and 973 program of China (Grant No. 2014CB340405). The authors conducted the research at Peking University and the University of Queensland. During the process of writing this book, we refer to the scientific researches of many scholars and experts. Hence, we would like to express our heartfelt thanks to them. We have been accompanied by a number of people who helped to shape the outcome of this book in one way or another. Hereby, we want to take this opportunity to convey our gratitude.

We thank Prof. Xiaofang Zhou, Prof. Shazia Sadiq, Dr. Kai Zheng, Dr. Sen Wang, Dr. Quoc Viet Hung Nguyen, and Weiqing Wang at the University of Queensland for their countless discussions and comments. Meanwhile, we want to thank Dr. Qi Liu at University of Science and Technology of China, Dr. Ling Chen from University of Technology Sydney, Dr. Yizhou Sun from Northeastern University, Prof. Wen-Chih Peng from National Chiao Tung University, A/Prof. Hao Wang from Institute of Software Chinese Academy of Science, Prof. Lei Zhao from Soochow University, and the editors Celine Chang and Jane Li at Springer for their valuable feedback to improve the presentation of this book.

Also, we worked with many great colleagues at the Data and Knowledge Engineering group, the University of Queensland, on a day-to-day basis. Thank you guys for the great work atmosphere, your sense of humor, and for making the frequent long working hours not feel that bad.

Last but not least, we would like to thank our own families for having our back. Thanks for all of your kind understanding and great support.

Contents

Chapter 1
Introduction

Abstract We first introduce the background and motivation for developing spatiotemporal recommendation in social media, and then analyze the significant research issues and challenges emerging in this field, including Temporal Context-Aware Recommendation, Spatial Recommendation for Out-of-Town Users, Location-based and Real-time Recommendation, and the Efficiency of Online Recommendation. We overview this book by listing our basic solution ideas for these problems and challenges. At the end of this chapter, we offer a rich overview of the related work and the relevant publications.

Keywords Social media · 4W Information layout · Semantically meaningful spatiotemporal data · Context-aware recommender system

1.1 Background

Over the past few years, one of the most important shifts in the digital world has been the move from the wide-open Web to semiclosed social media platforms. It is driven primarily by the rise of the mobile internet and smart phones. Social media services have been penetrated into all aspects in daily life. You wake up and check your messages on your WeChat or WhatsApp—that is one social media. During breakfast you browse Facebook, Twitter, and Digg—three more social media. During lunch time, you visit Yelp and look for restaurants, another social media. At the end of the day, you come home, make dinner while listening to Pandora and watch a movie on Youtube. You have spent the day on the social media, and you are not alone.

The recent advances in location-acquisition and wireless communication technologies such as GPS enable users to easily add a location dimension to traditional social networks (e.g., Twitter, Facebook, and Weibo) via a smartphone, and also fosters the growth of Location-based Social Networking services (LBSNs) such as Foursquare, Yelp, GeoLife, Meetup, and Google Place, where users can easily check-in at points of interests (e.g., restaurants, stores, hotels) and share their life experiences in the physical world via mobile devices, resulting in a "4W" (i.e., who, when, where and what) information layout, corresponding to four distinct

H. Yin and B. Cui, *Spatio-Temporal Recommendation in Social Media*,
SpringerBriefs in Computer Science, DOI 10.1007/978-981-10-0748-4_1

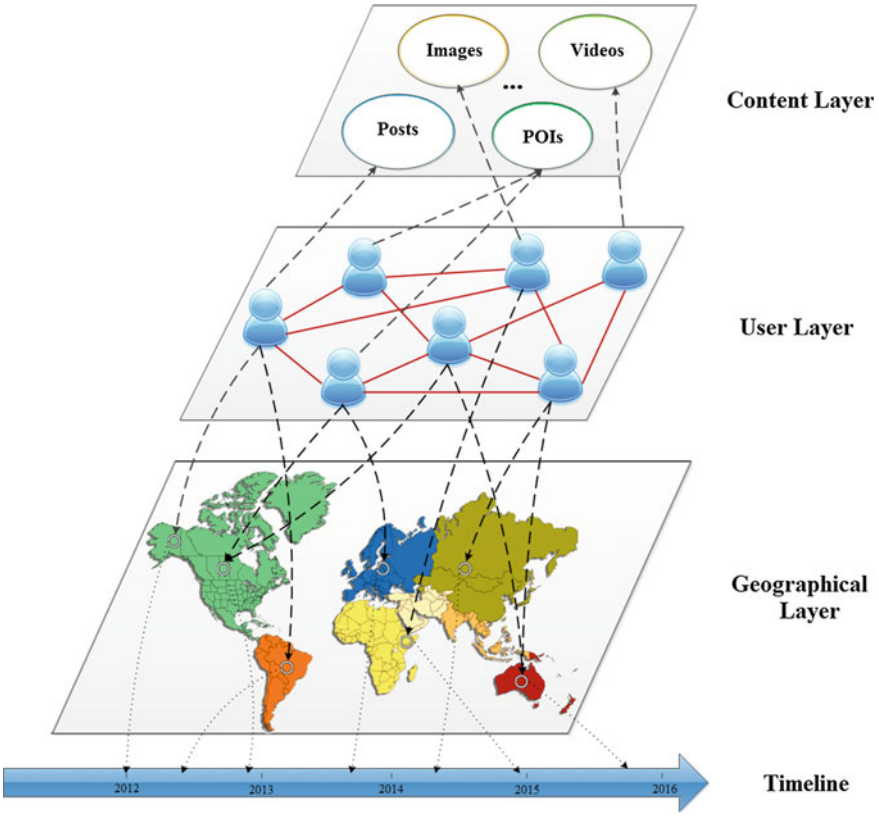

Fig. 1.1 The Information Layout of Social Media

information layers as shown in Fig. 1.1. The content information refers to tags, images, videos, comments, and posts generated or uploaded by users on social media, which represents the semantic information of users' activities. The user layer provides extensive knowledge about an individual, such as his/her demographics and social structure. The temporal and spatial layers provide rich contextual information of users' activities. Besides, the dimension of location also bridges the gap between online social networks and the physical world. Thus, the social media data in the mobile era captures the semantically meaningful snapshots of everyday lives, and provides unprecedented potential for user modeling. On the other hand, it is crucial to develop spatiotemporal recommendation services in social media, which help common users find their potentially preferred items, facilitate advertisers to launch targeted advertisements, and make social media platforms more attractive to users and advertisers.

1.2 The Research Issues and Challenges

In this section, we list the research issues and challenges in the spatiotemporal recommendation in social media.

- *Temporal Context-Aware Recommendation*: After investigating multiple social media systems, we observe that users' behaviors are dynamic since they are not only influenced by their personal interests (internal factor), but also by the dynamic social context (external factor), called temporal context. For example, given a user who prefers things of thriller and horror, he will not go to the haunted house or watch a horror movie on the Saint Valentine's Day. However, the data representing the user's personal interests and the temporal context respectively is mixed up together in social media. For example, some of my posts in Twitter are on my interests while others reflect the temporal context. Moreover, the two factors have different degrees of influence on users' behaviors on different social media platforms. Although on the same social media platform, the two factors have different influence strengths for different users. Thus, it is very challenging to simultaneously extract users' personal interests and discover the temporal contexts from the users' behavior data, and model their effect on users' decision-making in a unified way.
- *Recommendation for Out-of-Town Users*: With the rapid development of Location-based Social Networks (LBSNs) and Event-based Social Networks (EBSNs), spatial item recommendation has become an important means to help people discover attractive and interesting venues and events, especially when users travel out of town. However, this recommendation is very challenging compared to the traditional recommender systems. A user can visit only a limited number of spatial items, leading to a very sparse user-item matrix. Most of the items visited by a user are located within a short distance from where he/she lives, which makes it hard to recommend items when the user travels to a far away place (i.e., out of town), especially in a new city where she has no activity information. Moreover, users' interests and behavior patterns may vary dramatically across different geographical regions, due to different urban compositions and cultures.
- *Location-based and Real-time Recommendation*: As the time goes on and users' locations are changing, their interests and needs may change. In a certain short term, users' activities 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. In the long term, users' interests are intrinsically dynamic. This requires producing recommendation results in a real-time manner. However, the existing recommendation approaches or models [58, 63, 65, 67–69] developed for spatial item recommendation are incapable of supporting real-time recommendation due to the following reasons. First, most of them ignore users' real-time locations and the time information. Second, these methods assume that the individuals' interests are stable and ignore their dynamics in the long term. In reality, they are changing and evolving over time, as analyzed in [64]. For instance, users will naturally be

interested in visiting parenting-related items (e.g., the playground and amusement park) after they have a baby, and probably ignore their other interests. Accurately capturing this change in a real-time manner has been proved to be commercially very valuable since it indicates visiting and purchasing intents. Third, almost all existing recommendation models are based on batch learning, and it is difficult to apply them for large-scale users' activity or behavior data which arrives in a stream, since the batch learning algorithm needs to run through all the training data for many iterations, which is very time-consuming and infeasible.

- *The Efficiency of Online Recommendation*: Once the recommender models have been trained offline, given a query/user, the naive approach to produce online top-k recommendations is to first compute a ranking score for each item and then select k ones with highest ranking scores. However, when the number of available items becomes large, to produce a top-k ranked list using this brute-force method is very time-consuming and slow. Especially, to support real-time online recommendation, effective indexing techniques and smart top-k retrieval algorithms are needed to narrow down the search space of candidate items and find top-k results by scanning the minimum number of items.

1.3 Overview of the Book

The goal of this book is to develop a spatio-temporal user modeling framework to analyze the factors (e.g., personal interests, mobility patterns, local preferences, and temporal context) that influence users' behaviors and decision-making on social media, and then build an efficient intelligent recommender system to provide location-based, time-aware, real-time and personalized services by effectively overcoming the challenges proposed in Sect. 1.2. From a computing perspective, this book targets developing a computational foundation for mining, analyzing and modelling users' behavioral data, on top of which accurate and efficient recommendations can be achieved. The organization of this book is listed as follows.

- *Temporal Context-Aware Recommendation*: In Chap. 2, we analyze users' behaviors in multiple social media systems and design a latent class statistical mixture model, namely temporal context-aware mixture model (TCAM), to account for the intentions and preferences behind users' dynamic behaviors. Based on the observation that the behaviors of a user in social media systems are generally influenced by intrinsic interest as well as the temporal context (e.g., the public's attention at that time), TCAM simultaneously models the topics related to users' intrinsic interests and the topics related to temporal context and then combines the influences from the two factors to model user behaviors in a unified way. Extensive experiments have been conducted to evaluate the performance of TCAM on four real-world datasets crawled from different social media sites. The experimental results demonstrate the superiority of the TCAM models, compared with the

state-of-the-art competitor methods, by modeling user behaviors more precisely and making more effective recommendations.

- *Recommendation for Out-of-Town Users*: In Chap. 3, we propose LCA-LDA, a location-content-aware LDA model that provides a user with spatial item recommendations within the querying city by giving consideration to both personal interest and user interest drift. LCA-LDA learns the user's interests as a distribution over a set of latent topics, by mining both the co-occurrence patterns of spatial items and their content information (e.g., tags and categories). Exploiting content information of spatial items not only alleviates the data sparsity issue but also addresses the travel locality for out-of-town recommendation. The content of spatial items serves as the medium to transfer user interests learned from home town to unfamiliar regions. To adapt to user interest drift across regions, LCA-LDA learns local preferences/attractions of each region (e.g., city) from all the users' activity records within that specific region. We evaluate the performance of our LCA-LDA model on two large-scale real datasets, Douban-Event and Foursquare. The results show the superiority of LCA-LDA in recommending spatial items for users, especially when traveling to new cities.
- *Location-based and Real-time Recommendation*: In Chap. 4, we propose a unified probabilistic generative model, Topic-Region Model (TRM), to simultaneously discover the semantic, temporal cyclic and spatial mobility patterns of users' check-in activities, and to model their joint effect on users' decision-making. Thus, TRM is capable of providing accurate recommendations for mobile users according to their real-time locations and querying time. To support real-time recommendation, we further extend the TRM model to an online learning model TRM-Online to track changing user interests and speed up the model training. We conduct extensive experiments to evaluate the performance of our proposals on two real-world datasets including recommendation effectiveness, overcoming cold-start problem and model training efficiency. The experimental results demonstrate the superiority of our TRM models, especially the TRM-Online, compared with the state-of-the-art competing methods, by making more effective and efficient mobile recommendations.
- *The Efficiency of Online Recommendation*: Based on the spatio-temporal recommender models developed in the previous three chapters, the top-k recommendation task can be reduced to an simple task of finding the top-k items with the maximum dot-products for the query/user vector over the set of item vectors. In Chap. 5, we build effective multi-dimensional index structures and inverted index structures to manage the item vectors, and develop three efficient top-k retrieval algorithms to speed up the online spatio-temporal recommendation. These three algorithms are Metric-Tree based search algorithm (MT), Threshold-based algorithm (TA) and Attribute Pruning-based algorithm (AP). MT and TA focus on pruning item search space while AP aims to prune attribute space. To evaluate the performance of our developed techniques, we conduct extensive experiments on both real-world and large-scale synthetic datasets. The experimental results show that MT, TA, and AP can achieve superior performance under different data dimensionality.

1.4 Literature and Research Review

In this section, we review related literatures, including traditional context-aware recommendation, temporal recommendation, and spatial recommendation.

1.4.1 Traditional Context-Aware Recommendation

Collaborative filtering and content-based filtering techniques are two widely adopted approaches for recommender systems [2]. Both of them discover users' personal interests and utilize these discovered interests to find relevant items. Collaborative filtering techniques [16, 53] automatically suggest relevant items for a given user by referencing item rating information from other taste-similar users. The content-based recommendation [52] is based on the assumption that descriptive features of an item tell much about a user's preference for an item. Recommender systems using pure collaborative filtering approaches tend to fail when little knowledge about the user is known or when no one has interests similar to the user's. Although the content-based method is capable of coping with the lack of knowledge, it fails to account for community endorsement. As a result, a certain amount of research has focused on combining the advantages of both collaborative filtering and content-based methods [11, 36, 49].

The importance of contextual information has been recognized by researchers and practitioners in many disciplines, including e-commerce personalization, information retrieval, ubiquitous and mobile computing, data mining, marketing, and management. Because of the influential role of contextual information to recommendations, Context-Aware Recommender Systems (CARS) have recently attracted the high attention of researchers. Currently, there are basically three paradigms to make context-aware recommendations including the contextual pre-filtering, contextual post-filtering and contextual modeling [3].

Pre-filtering approaches typically apply certain context dependent criteria to filter the list of items at first, and then select appropriate items to the given contextual condition. As a consequence, only filtered items can be considered for further recommendations. Adomavicius et al. [1] presented a context-aware approach based on the multidimensional reduction. They extended traditional collaborative filtering recommendation methods by adding extra dimensions that represent contextual information. The item splitting [7] was another pre-filtering approach based on the idea that items experienced in two alternative contextual conditions are "split" into two items. This means that the ratings of a split item, e.g., a place to visit, are assigned (split) to two new fictitious items representing for instance the place in summer and the same place in winter. Lately, user splitting, and user-item splitting [8] were proposed to run splitting in different manners. Splitting methods were reported as the most efficient and effective context-aware algorithms [7, 8]. While in post-filtering approaches, contextual information is usually used after some certain

traditional recommendation approaches have been deployed. Items that are not suited in the given condition should be neglected in the final recommendation. The experimental comparison between different filtering methods was reported in [8]. They found that there are no clear winners among these approaches. The pre-filtering and post-filtering approaches all attempt to convert the problem of context-aware recommendation into the traditional recommendation. They all try to filter out preferences that do not match the current condition. On the contrary, contextual modeling methods model the contextual information directly into recommendation algorithms. For instance, Oku et al. [48] utilized the SVM classifier for contextual dependent recommendations. Lately, the well-known matrix factorization technique, which is most popular for traditional recommender systems, has been adopted to CARS [6]. In addition, Karatzoglou et al. [35] proposed a multiverse factorization model, in which the preference matrix was modeled as a user-item-context n-dimensional tensor. Tensor factorization technique was then applied to CARSs. These factorization methods suffer from sparsity and scalability issues in training parameters [6, 51, 54]. One possible solution to alleviate the scalability issue is to partition the original matrix before applying any factorization models [79].

All existing CAR methods treat multiple contexts equally and ignore the fact that different contexts (e.g., social context, spatial context, and temporal context) have different characteristics. One-fit-all method is not a good solution to exploit multiple context information. Different strategies should be adopted for different contexts. In this book, we study the unique characteristics of temporal context and spatial context information, as the spatial and temporal context information are more easier to automatically collect from social media systems and play a more important role in the era of mobile internet and smart phones, compared with other contextual information. Based on the characteristics of the temporal and spatial contextual information, different strategies are proposed to leverage them to improve the recommendation results. Besides, all existing context-aware recommendation methods assume that the contextual information is discrete, while the methods developed in this book can deal with both discrete and continuous contextual information.

1.4.2 Temporal Recommendation

Being different from traditional recommendation, temporal recommendation takes the time factor into account and aims to provide dynamic recommendation. Many successful temporal collaborative filtering methods are based on latent factor models. For example, the Netflix award winning algorithm *timeSVD++* [38] assumes that the latent features consist of components that evolve over time and a dedicated bias for each user at each specific time point. This model can effectively capture local changes to user preferences which the authors claim to be vital for improving performance. Xiong et al. proposed a Bayesian probabilistic tensor factorization model (BPTF) in [60]. BPTF represents users, items and time in a shared low-dimensional space, and predicts the rating score that a user u will assign to item v at time t using the inner

product of their latent representations. Demonstrated by the experimental results on Netflix data, both BPTF and time SVD++ perform well on the rating prediction task because they incorporate time effects into models. One main disadvantage with these models is that the learnt latent low-dimensional space is difficult to interpret. Recently, Liu et al. [42] addressed a new problem—online evolutionary collaborative filtering, which tracks user interests over time for the purpose of making timely recommendations. They extended the widely used neighborhood based algorithms by incorporating temporal information and developed an incremental algorithm for updating neighborhood similarities with new data. However, most of the existing temporal recommendation models [38, 42, 60] are designed for the task of rating prediction rather than top-k recommendation. Diao et al. [23] assumed that user behaviors are influenced by both user interests and global topic trends, and proposed mixture latent topic models to capture these factors. But, these models make use of one set of shared topics to model two factors. The estimated topics are confusing and difficult to interpret, which causes the recommendation results to degenerate. To improve the topic discovery process, Yin et al. [66] recently proposed a unified model to detect both stable and temporal topics simultaneously from social media data.

1.4.3 Spatial Item Recommendation

Spatial item recommendation, consisting of POI recommendation and event recommendation [45, 74], 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 [77, 78]. Because of the unavailability of content information associated with POIs, spatial and temporal patterns are commonly integrated into collaborative filtering methods to make POI recommendation. 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 geo-social, temporal, and semantic information associated with users' activities.

Geo-Social Information. Many recent studies [19, 20, 26, 62, 73, 80] showed that there is a strong correlation between user check-in activities and geographical distance as well as social connections, so most of current POI recommendation work mainly focuses on leveraging the geographical and social influences to improve recommendation accuracy. For example, Ye et al. [62] 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. [17] investigated the geographical influence through combining a multicenter Gaussian model, matrix factorization and social influence together for location recommendation. Lian et al. [39] incorporated spatial clustering phenomenon

resulted by geographical influence into a weighted matrix factorization framework to deal with the challenge from matrix sparsity. However, all of them do not consider the current location of the user. Thus, no matter whether the user is located in the home town or traveling out of town, they will recommend the same POIs to the user. In light of this, Ference et al. [26] designed a collaborative recommendation framework which 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 recommendation with temporal effect mainly leverage temporal cyclic patterns and temporal sequential patterns on LBSNs. Gao et al. [28] investigated the temporal cyclic patterns of user check-ins in terms of temporal non-uniformness and temporal consecutiveness. Yuan et al. [71] incorporated the temporal cyclic information into a user-based collaborative filtering framework for time-aware POI recommendation. Cheng et al. [18] introduced the task of successive personalized POI recommendation in LBSNs by embedding the temporal sequential patterns.

As described above, 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 the out-of-town recommendation. Most of the above work assumed that users are in their home towns, they did not consider users' real-time locations, nor their interest drift across geographical regions.

Semantic Information. Most recently, researchers explored the content information of POIs to alleviate the problem of data sparsity. Hu et al. [33] proposed a spatial topic model for POI recommendation considering both spatial aspect and textual aspect of user posts from Twitter. Liu et al. [41] studied the effect of POI-associated tags for POI recommendation with an aggregated LDA and matrix factorization method. Yin et al. [67] exploited both personal interests and local preferences based on the contents associated with spatial items. Gao et al. [29] and Zhao et al. [76] studied both POI-associated contents and user sentiment information into POI recommendation. However, all of them do not consider the time information associated with the contents of POIs.

Although some recent literatures [10, 47] used classification-based method to predict the next place a user will move by extracting multiple features from users' movement history, 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.

1.4.4 Real-Time Recommendation

Recently, real-time recommendation has attracted a lot of attention from both industry and academia [15, 24, 34, 44, 56, 57]. 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 recommendation, various incremental online matrix factorization or collaborative filtering models are proposed [15, 24, 34, 44, 56, 57], and the efficiency of matrix factorization models has also been extensively studied [30, 70]. 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 [26, 67]. Meanwhile, the developed online model training techniques for matrix factorization cannot be used to train our developed spatio-temporal recommender models. This book focuses on the efficiency of LDA-like model training.

Most existing researches on LDA-like model utilize various inference algorithms, such as variational Bayesian [13, 27], Gibbs sampling [31], expectation propagation [46] and belief propagation [72] to obtain the parameters. Unfortunately, most of them are batch algorithms. Recently, a host of online learning methods have been developed for topic models. Some of them focus on modeling large amount of documents efficiently based on LDA. Typical researches include TM-LDA [59], On-Line LDA [4] 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 work tries to improve inference algorithms themselves. Banerjee et al. [9] presented online variants of vMF, EDCM, and LDA. Their experiments illustrated faster speed and better performance on real-world streaming text. Homan et al. developed an online variational Bayesian algorithm [32] 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. also proposed an online version algorithm [14] using Gibbs sampling. Yao et al. compared several batch methods mentioned above, and introduced a new algorithm, SparseLDA [61], to accelerate learning process.

In summary, a large number of prior works have made great effort on designing appropriate online algorithms for LDA to process documents. However, less work focuses on the users' activity (e.g., check-ins) stream and tracking changing user interests and mobility patterns. In this book, we propose a novel online learning model TRM-Online to process check-in stream and track changing user interests and mobility patterns for supporting mobile recommendation in a real-time manner.

1.4.5 Online Recommendation Efficiency

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 w.r.t. a query is computed as the inner product between their vectors. To the best of our knowledge, Ram and Gray [50] were the first to propose and address the problem of fast maximum inner-product search. They proposed to organize the item vectors in a cone ball tree, 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 cone ball tree itself is constructed by repeatedly splitting the set of item vectors into two partitions (based on Euclidean distances). In the subsequent work [21], the cone ball tree is replaced by a cover tree [12]. 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 [43], the speedup techniques using pure tree structures suffer from the curse of the dimensionality, and their time cost is $O(A^{12})$ (A is the dimension of items). Our experimental study also demonstrates that tree indexing-based algorithms are not effective for items with high dimensionality in practice.

TA-Based Algorithms. Yin et al. [64, 67] extended the popular threshold algorithm (TA) [25] 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 ($A < 200$). 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 down the efficiency of TA when the dimensionality is high.

Approximate Algorithms. Approximate methods for fast retrieval of top-k recommendation have also been studied in the literature. We categorize them as non-hashing methods and hashing methods. We first review non-hashing methods. In [40], Linden et al. discussed the idea of item-space partitioning, which makes recommendation from some subsets of all items. They concluded such a naive strategy would produce recommendations of low quality. User clustering was adopted in [22, 37] so that similar users share the same recommendation results. Although this strategy essentially reduces the user space, it also degrades the performances of personalized recommendation. Yin et al. [67] extended TA to a ρ-approximation algorithm to speed up the online recommendation 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 [55] or modified PCA-trees [5]. An alternative approach is taken by [75], 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.

References

1. Adomavicius, G., Sankaranarayanan, R., Sen, S., Tuzhilin, A.: Incorporating contextual information in recommender systems using a multidimensional approach. ACM Trans. Inf. Syst. **23**(1), 103–145 (2005)
2. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans. Knowl. Data Eng. **17**(6), 734–749 (2005)
3. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans. Knowl. Data Eng. **17**(6), 734–749 (2005)
4. AlSumait, L., Barbara, D., Domeniconi, C.: On-line lda: adaptive topic models for mining text streams with applications to topic detection and tracking. In: Proceedings of ICDM, pp. 3–12 (2008)
5. Bachrach, Y., Finkelstein, Y., Gilad-Bachrach, R., Katzir, L., Koenigstein, N., Nice, N., Paquet, U.: Speeding up the xbox recommender system using a euclidean transformation for inner-product spaces. In Proceedings of the RecSys, pp. 257–264 (2014)
6. Baltrunas, L., Ludwig, B., Ricci, F.: Matrix factorization techniques for context aware recommendation. In Proceedings of RecSys, pp. 301–304 (2011)
7. Baltrunas, L., Ricci, F.: Context-based splitting of item ratings in collaborative filtering. In Proceedings of the RecSys, pp. 245–248 (2009)
8. Baltrunas, L., Ricci, F.: Experimental evaluation of context-dependent collaborative filtering using item splitting. User Model. User-Adap. Interact. **24**(1–2), 7–34 (2014)
9. Banerjee, A., Basu, S.: Topic models over text streams: A study of batch and online unsupervised learning. SDM **7**, 437–442 (2007)
10. Baraglia, R., Muntean, C. I., Nardini, F. M., Silvestri. F.: Learnext: learning to predict tourists movements. In Proceedings of CIKM, pp. 751–756 (2013)
11. Basilico, J., Hofmann, T.: Unifying collaborative and content-based filtering. In Proceedings of ICML (2004)
12. Beygelzimer, A., Kakade, S., Langford, J.: Cover trees for nearest neighbor. In Proceedings of ICML, pp. 97–104 (2006)
13. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. J. Mach. Learn. Res. **3**, 993–1022 (2003)
14. Canini, k. R., Shi, L., Griffiths, T. L.: Online inference of topics with latent dirichlet allocation. In Proceedings of AISTATS, pp. 65–72 (2009)
15. Chandramouli, B., Levandoski, J. J., Eldawy, A., Mokbel, M. F.: Streamrec: a real-time recommender system. In Proceedings of SIGMOD, pp. 1243–1246 (2011)
16. Chen, W.-Y., Chu, J.-C., Luan, J., Bai, H., Wang, Y., Chang. E. Y.: Collaborative filtering for orkut communities: discovery of user latent behavior. In Proceedings of WWW, pp. 681–690 (2009)
17. Cheng, C., Yang, H., King, I., Lyu. M. R.: Fused matrix factorization with geographical and social influence in location-based social networks. In Proceedings of AAAI (2012)
18. Cheng, C., Yang, H., Lyu, M. R., King, I.: Where you like to go next: successive point-of-interest recommendation. In Proceedings of IJCAI, pp. 2605–2611 (2013)
19. Cheng, Z., Caverlee, J., Lee, K., Sui, D. Z.: Exploring millions of footprints in location sharing services. In Proceedings of ICWSM (2011)
20. Cho, E., Myers, S. A., Leskovec, J.: Friendship and mobility: user movement in location-based social networks. In Proceedings of KDD, pp.1082–1090 (2011)
21. Curtin, R. R., Gray, A. G., Ram, P., Alexander, R. R. C. P. R., Gray, G.: Fast exact max-kernel search. In Proceedings of SDM, pp.1–9 (2013)
22. Das, A. S., Datar, M., Garg, A., Rajaram, S.: Google news personalization: scalable online collaborative filtering. In Proceedings of WWW, pp. 271–280 (2007)
23. Diao, Q., Jiang, J., Zhu, F., Lim, E.-P.: Finding bursty topics from microblogs. In Proceedings of ACL, pp. 536–544 (2012)

24. Diaz-Aviles, E., Drumond, L., Schmidt-Thieme, L., Nejdl. W.: Real-time top-n recommendation in social streams. In Proceedings of RecSys, pp. 59–66 (2012)
25. Fagin, R., Lotem, A., Naor, M.: Optimal aggregation algorithms for middleware. In Proceedings of PODS, pp. 102–113 (2001)
26. Ference, G., Ye, M., Lee, W.-C.: Location recommendation for out-of-town users in location-based social networks. In Proceedings of CIKM, pp. 721–726 (2013)
27. Foulds, J., Boyles, L., DuBois, C., Smyth, P., Welling, M.: Stochastic collapsed variational bayesian inference for latent dirichlet allocation. In Proceedings of KDD, pp. 446–454 (2013)
28. Gao, H., Tang, J., Hu, X., Liu, H.: Exploring temporal effects for location recommendation on location-based social networks. In Proceedings of RecSys, pp. 93–100 (2013)
29. Gao, H., Tang, J., Hu, X., Liu, H.: Content-aware point of interest recommendation on location-based social networks. In Proceedings of AAAI (2015)
30. Gemulla, R., Nijkamp, E., Haas, P. J., Sismanis, Y.: Large-scale matrix factorization with distributed stochastic gradient descent. In Proceedings of KDD, pp.69–77 (2011)
31. Griffiths, T.L., Steyvers, M.: Finding scientific topics. Proc. Natl. Acad. Sci. **101**(suppl 1), 5228–5235 (2004)
32. Hoffman, M., Bach, F. R., Blei, D. M.: Online learning for latent dirichlet allocation. In Proceedings of NIPS, pp. 856–864 (2010)
33. Hu, B., Ester, M.: Spatial topic modeling in online social media for location recommendation. In Proceedings of RecSys, pp. 25–32 (2013)
34. Huang, Y., Cui, B., Zhang, W., Jiang, J., Xu, Y.: Tencentrec: real-time stream recommendation in practice. In Proceedings of the IGMOD, pp. 227–238 (2015)
35. Karatzoglou, A., Amatriain, X., Baltrunas, L., Oliver. N.: Multiverse recommendation: N-dimensional tensor factorization for context-aware collaborative filtering. In Proceedings of the RecSys, pp. 79–86 (2010)
36. Kim, B.M., Li, Q., Park, C.S., Kim, S.G., Kim, J.Y.: A new approach for combining content-based and collaborative filters. J. Intell. Inf. Syst. **27**(1), 79–91 (2006)
37. Koenigstein, N., Ram, P., Shavitt, Y.: Efficient retrieval of recommendations in a matrix factorization framework. In Proceedings of the CIKM, pp. 535–544 (2012)
38. Koren, Y.: Collaborative filtering with temporal dynamics. In Proceedings of the KDD, pp. 447–456 (2009)
39. Lian, D., Zhao, C., Xie, X., Sun, G., Chen, v., Rui, Y.: Geomf: joint geographical modeling and matrix factorization for point-of-interest recommendation. In Proceedings of the KDD, pp. 831–840 (2014)
40. Linden, G., Smith, B., York, J.: Amazon.com recommendations: item-to-item collaborative filtering. IEEE Internet Comput. **7**(1), 76–80 (2003)
41. Liu, B., Xiong, H.: Point-of-interest recommendation in location based social networks with topic and location awareness. In Proceedings of the SDM, pp. 396–404 (2013)
42. Liu, N. N., Zhao, M., Xiang, E., Yang, Q.: Online evolutionary collaborative filtering. In Proceedings of the RecSys, pp. 95–102 (2010)
43. Liu, T., Moore, A. W., Yang, K., Gray, A. G.: An investigation of practical approximate nearest neighbor algorithms. In Proceedings of the NIPS, pp. 825–832 (2004)
44. Lommatzsch, A., Albayrak, S.: Real-time recommendations for user-item streams. In Proceedings of the SAC, pp. 1039–1046 (2015)
45. Macedo, A. Q., Marinho, L. B., Santos, R. L.: Context-aware event recommendation in event-based social networks. In Proceedings of the RecSys, pp. 123–130 (2015)
46. Minka, T., Lafferty, J.: Expectation-propagation for the generative aspect model. In Proceedings of the UAI, pp. 352–359 (2002)
47. Noulas, A., Scellato, S., Lathia, N., Mascolo, C.: Mining user mobility features for next place prediction in location-based services. In Proceedings of the ICDM, pp. 1038–1043 (2012)
48. Oku, K., Nakajima, S., Miyazaki, J., Uemura, S.: Context-aware svm for context-dependent information recommendation. In Proceedings of the MDM, p. 109 (2006)
49. Popescul, A., Pennock, D. M., Lawrence, S.: Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. In Proceedings of the UAI, pp. 437–444 (2001)

50. Ram, P., Gray, A. G.: Maximum inner-product search using cone trees. In Proceedings of the KDD, pp. 931–939 (2012)
51. Rendle, S., Gantner, Z., Freudenthaler, C., Schmidt-Thieme, L.: Fast context-aware recommendations with factorization machines. In Proceedings of the SIGIR, pp. 635–644 (2011)
52. Ricci, F., Rokach, L., Shapira, B., Kantor, P. B.: Recommender Systems Handbook. Springer, New York (2010)
53. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In Proceedings of the WWW, pp. 285–295 (2001)
54. Shi, Y., Karatzoglou, A., Baltrunas, L., Larson, M., Hanjalic, A., Oliver, N.: Tfmap: optimizing map for top-n context-aware recommendation. In Proceedings of the SIGIR, pp. 155–164 (2012)
55. Shrivastava, A., Li, P.: Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). In Proceedings of the NIPS, pp. 2321–2329 (2014)
56. Silva, J., Carin, L.: Active learning for online bayesian matrix factorization. In Proceedings of the KDD, pp.325–333 (2012)
57. Vinagre, J., Jorge, A. M., Gama, J.: Fast incremental matrix factorization for recommendation with positive-only feedback. In Proceedings of the UMAP, pp. 459–470 (2014)
58. Wang, W., Yin, H., Chen, L., Sun, Y., Sadiq, S., Zhou, X.: Geo-sage: a geographical sparse additive generative model for spatial item recommendation. In Proceedings of the KDD, pp.1255–1264 (2015)
59. Wang, Y., Agichtein, E., Benzi, M.: Tm-lda: efficient online modeling of latent topic transitions in social media. In Proceedings of the KDD, pp. 123–131 (2012)
60. Xiong, L., Chen, X., Huang, T.-K., Schneider, J. G., Carbonell, J. G.: Temporal collaborative filtering with bayesian probabilistic tensor factorization. In Proceedings of the SDM, pp. 211–222 (2010)
61. Yao, L., Mimno, D., McCallum, A.: Efficient methods for topic model inference on streaming document collections. In Proceedings of the KDD, pp. 937–946 (2009)
62. Ye, M., Yin, P., Lee, W.-C., Lee, D.-L.: Exploiting geographical influence for collaborative point-of-interest recommendation. In Proceedings of the SIGIR, pp. 325–334 (2011)
63. Yin, H., Cui, B., Chen, L., Hu, Z., Zhang, C.: Modeling location-based user rating profiles for personalized recommendation. ACM Trans. Knowl. Discov. Data, **9**(3):19:1–19:41 (2015)
64. Yin, H., Cui, B., Chen, L., Hu, Z., Zhou, X.: Dynamic user modeling in social media systems. ACM Trans. Inf. Syst. **33**(3):10:1–10:44 (2015)
65. Yin, H., Cui, B., Huang, Z., Wang, W., Wu, X., Zhou, X.: Joint modeling of users' interests and mobility patterns for point-of-interest recommendation. In Proceedings of the MM, pp. 819–822 (2015)
66. Yin, H., Cui, B., Lu, H., Huang, Y., Yao, J.: A unified model for stable and temporal topic detection on social platform. In Proceedings of the ICDE (2013)
67. Yin, H., Cui, B., Sun, Y., Hu, Z., Chen, L.: Lcars: a spatial item recommender system. ACM Trans. Inf. Syst. **32**(3):11:1–11:37 (2014)
68. Yin, H., Sun, Y., Cui, B., Hu, Z., Chen, L.: Lcars: a location-content-aware recommender system. In Proceedings of the KDD, pp. 221–229 (2013)
69. Yin, H., Zhou, X., Shao, Y., Wang, H., Sadiq, S.: Joint modeling of user check-in behaviors for point-of-interest recommendation. In Proceedings of the CIKM, pp. 1631–1640 (2015)
70. Yu, H.-F., Hsieh, C.-J., Si, S., Dhillon, I. S.: Scalable coordinate descent approaches to parallel matrix factorization for recommender systems. In Proceedings of the ICDM, pp. 765–774 (2012)
71. Yuan, Q., Cong, G., Ma, Z., Sun, A., Thalmann, N. M.: Time-aware point-of-interest recommendation. In Proceedings of the SIGIR, pp.363–372 (2013)
72. Zeng, J., Cheung, W., Liu, J.: Learning topic models by belief propagation. IEEE Trans. Pattern Anal. Mach. Intell. **35**(5), 1121–1134 (2013)
73. Zhang, C., Shou, L., Chen, K., Chen, G., Bei, Y.: Evaluating geo-social influence in location-based social networks. In Proceedings of the CIKM, pp. 1442–1451 (2012)

74. Zhang, W., Wang, J.: A collective bayesian poisson factorization model for cold-start local event recommendation. In Proceedings of the KDD, pp. 1455–1464 (2015)
75. Zhang, Z., Wang, Q., Ruan, L., Si, L.: Preference preserving hashing for efficient recommendation. In Proceedings of the SIGIR, pp. 183–192 (2014)
76. Zhao, K., Cong, G., Yuan, Q., Zhu, K.: Sar: a sentiment-aspect-region model for user preference analysis in geo-tagged reviews. In Proceedings of the ICDE (2015)
77. Zheng, Y., Xie, X.: Learning travel recommendations from user-generated gps traces. ACM Trans. Intell. Syst. Technol. **2**(1):2:1–2:29 (2011)
78. Zheng, Y., Zhang, L., Xie, X., Ma, W.-Y.: Mining interesting locations and travel sequences from gps trajectories. In Proceedings of the WWW, pp. 791–800 (2009)
79. Zhong, E., Fan, W., Yang, Q.: Contextual collaborative filtering via hierarchical matrix factorization. In Proceedings of the SDM, pp. 744–755 (2012)
80. Zhu, W.-Y., Peng, W.-C., Chen, L.-J., Zheng, K., Zhou, X.: Modeling user mobility for location promotion in location-based social networks. In Proceedings of the KDD, pp. 1573–1582 (2015)

Chapter 2
Temporal Context-Aware Recommendation

Abstract Users' behaviors in social media systems are generally influenced by intrinsic interest as well as the temporal context (e.g., the public's attention at that time). In this chapter, we focus on analyzing user behaviors in social media systems and designing a latent class statistical mixture model, named *temporal context-aware mixture model* (TCAM), to account for the intentions and preferences behind user behaviors. TCAM simultaneously models the topics related to users' intrinsic interests and the topics related to temporal context, and then combines the influences from the two factors to model user behaviors in a unified way. To further improve the performance of TCAM, an item-weighting scheme is proposed to enable TCAM to favor items that better represent topics related to user interests and topics related to temporal context, respectively. Extensive experiments have been conducted to evaluate the performance of TCAM on four real-world datasets crawled from different social media sites. The experimental results demonstrate the superiority of the TCAM models.

Keywords Dynamic user behavior · Temporal recommendation · Temporal context modeling · Entropy filtering

2.1 Introduction

With the rising popularity of social media, a better understanding of users' rating behaviors[1] is of great importance for the design of many applications, such as personalized recommendation, information filtering, behavioral targeting, and computational advertising. Research efforts [12, 15] have been undertaken to model users' interests to help them find interesting items by analyzing their historical behaviors. However, existing work [12, 15, 17] simply assumes that users prefer items based on their intrinsic interests, which may not be accurate in many social application scenarios. For example, when choosing a book to read or a movie to watch, the users are likely to prefer books/movies that interest them. In contrast, when selecting news

[1] We use the term "rating behavior" to denote general user actions on items in social media systems, such as rating and viewing.

H. Yin and B. Cui, *Spatio-Temporal Recommendation in Social Media*,
SpringerBriefs in Computer Science, DOI 10.1007/978-981-10-0748-4_2

to read or users to follow in a social network (e.g., Twitter), it is most likely that users will be attracted, respectively, by breaking news or famous users who are followed by the general public [4, 10, 20]. Therefore, users' rating behaviors on items may not necessarily indicate users' intrinsic interests. New models are desired to better account for user behaviors in social medias to learn user preferences more precisely.

After investigating multiple social media systems, we observe that user rating behaviors are generally influenced by two factors: *the intrinsic interest of the user* and *the attention of the general public*. While the user's intrinsic interest is relatively stable, the attention of the general public changes from time to time; for example, the hot topics on a microblogging site evolve over time. Hence, in our work, we refer to the attention of the public during a particular time period as *temporal context*.

The two factors have different degrees of influence on user rating behaviors for different types of social media platforms as a result of the different characteristics (e.g., life cycles and updating rates) of various types of social media items. For instance, news is a type of time-sensitive item with a short life cycle—few people want to read outdated news; while the life cycle of movies is relatively longer, with many classic old movies being highly ranked in the popularity list. For time-sensitive social media items, users are more easily influenced by the temporal context, whereas they tend to make decisions based on their intrinsic interests when choosing less time-sensitive items such as books and movies.

To model user rating behaviors in social media systems, therefore, it is critical to identify users' intrinsic interests as well as the temporal context (i.e., the attention of the general public during a particular time period). Moreover, it is essential to model the influence degrees of the two factors in different social media systems.

To this end, we proposed TCAM to mimic user rating behaviors in a process of decision making in [21, 22]. As shown in Fig. 2.1, TCAM is a latent class statistical mixture model that simultaneously models the topics [1, 8] related to users' intrinsic interests and the topics related to temporal context, and then combines the influences from the user interest and the temporal context to model user behaviors in a unified manner. Specifically, the model discovers (1) users' personal interest distribution over a set of latent topics; (2) the temporal context distribution over a set of latent topics; (3) an item generative distribution for each latent topic; and (4) the mixing weights that represent the influence probabilities of users' personal interest and the temporal context. It is worth mentioning that the set of latent topics used to model user interest is different from the topics used to model the temporal context. The former are called *user-oriented topics* and the latter are referred to as *time-oriented topics*.

The generative process of user rating behaviors in TCAM is briefly illustrated as follows. Suppose a user u selects an item v in a time interval t. TCAM first tosses a coin, based on the influence probabilities of the two factors, to decide whether this behavior results from the influence of the user's personal interest or the influence of the temporal context. If it results from the influence of the user's personal interest, TCAM chooses a *user-oriented topic* for u based on the user's intrinsic interest (with a certain probability). The selected topic in turn generates an item v following on from the topic's item generative distribution. Otherwise, if the influence from the

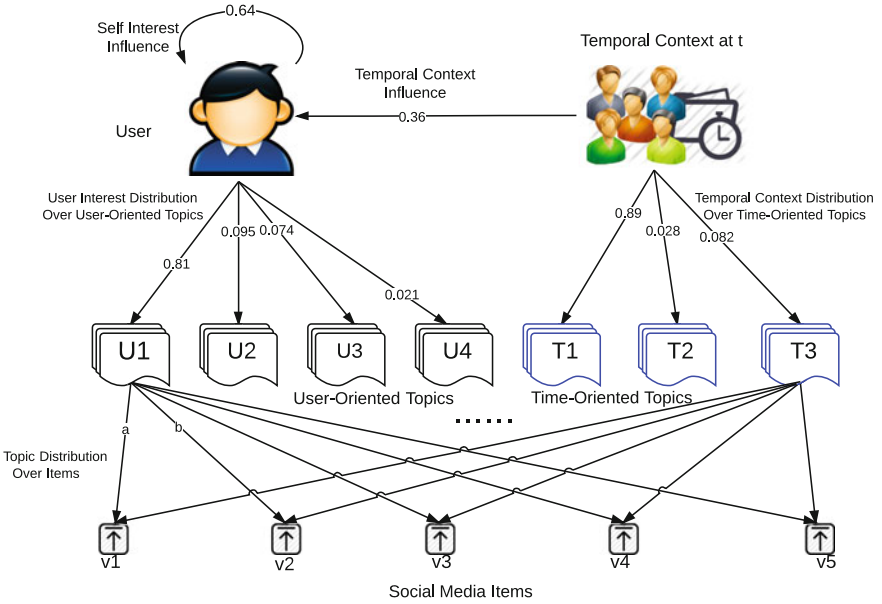

Fig. 2.1 An example of TCAM model [22] © 2015 Association for Computing Machinery, Inc. Reprinted by permission

temporal context is sampled, TCAM chooses a *time-oriented topic* according to the general public's interest during t, which in turn generates an item v.

Similar to traditional topic models where popular words in a document corpus are usually ranked high in each topic [3, 4], popular social media items tend to be estimated as having high generation probability by TCAM, which impairs the quality of the discovered user-oriented topics and time-oriented topics. User-oriented topics are supposed to capture user intrinsic interests, but a popular item favored by many users conveys less information about a user's intrinsic interest than an item favored by few users (i.e., a salient item) [23]. Similarly, a popular item constantly favored by users cannot well represent a time-oriented topic because the public's attentions change over time. Hence, to improve the performance of TCAM, we devise an item-weighting scheme to promote the importance of salient items and bursty items, which enhances the quality of the underlying topics detected by TCAM.

The remainder of this chapter is organized as follows. Section 2.2 details the TCAM. We deploy TCAM to temporal recommendation in Sect. 2.3. We carry out extensive experiments and report the experimental results in Sect. 2.4, and conclude the chapter in Sect. 2.5.

2.2 User Rating Behavior Modeling

In this section, we first introduce relevant definitions and notations used throughout this chapter. We then present the novel temporal context-aware mixture model for modeling user rating behaviors in social media systems.

2.2.1 Notations and Definitions

The notations used in this chapter are summarized in Table 2.1.

Definition 2.1 (*User Rating*) A user rating is a triple (u, t, v) that denotes a rating behavior (e.g., purchasing, clicking and tagging) made by user u on item v during time interval t.

Definition 2.2 (*User Document*) Given a user u, the user document, D_u, is a set of pairs $\{(v,\ t)\}$ representing the rating behaviors on items during different time intervals made by u.

Definition 2.3 (*Rating Cuboid*) A rating cuboid $\boldsymbol{C}$ is an $N \times T \times V$ cuboid, where N is the number of users, T is the number of time intervals and V is the number of items. A cell indexed by $(u,\ t,\ v)$ stores the rating score that user u assigned to item v during time interval t.

Table 2.1 Notations used in this model

Symbol	Description
u, t, v	User u, time interval t, item v
N, T, V	Number of users, time intervals, and items
M_u	Number of items rated by user u
λ_u	The mixing weight specific to user u
K_1	Number of user-oriented topics
$\theta_{u,z}$	Probability that user-oriented topic z is chosen by user u
θ_u	Intrinsic interest of user u denoted by $\theta_u = \{\theta_{u,z}\}_{z=1}^{K_1}$
ϕ_z	Item proportions of user-oriented topic z, denoted by $\phi_z = \{\phi_{z,v}\}_{v=1}^{V}$
$\phi_{z,v}$	Probability that item v is generated by user-oriented topic z
K_2	Number of time-oriented topics
θ'_t	The temporal context during time interval t denoted by $\theta'_t = \{\theta'_{t,x}\}_{x=1}^{K_2}$
$\theta'_{t,x}$	Probability that time-oriented topic x is generated by time interval t
ϕ'_x	Item proportions of time-oriented topic x denoted by $\phi'_x = \{\phi'_{x,v}\}_{v=1}^{V}$
$\phi'_{x,v}$	Probability that item v is generated by time-oriented topic x

User actions on items, such as tagging, downloading, purchasing, and clicking, can be represented as a user rating. Either explicit feedback or implicit feedback can be used to compute the value of rating score. For example, given a user u who frequently uses a tag v during time interval t, the usage frequency can be used as the rating score to reflect the user's preference on the tag during that time period.

Definition 2.4 (*Topic*) Given a collection of items $\boldsymbol{I} = \{v_i\}_{i=1}^{V}$, a topic z is represented by a topic model ϕ_z, which is a multinomial distribution over items $\phi_z = \{P(v_i|\phi_z)$ or $\phi_{z,v_i}\}_{i=1}^{V}$.

To illustrate the semantic meaning of a topic, we choose top-k items that have the highest probability under the topic, as shown in Fig. 2.2. In our work, we distinguish between **user-oriented topics** ϕ_z and **time-oriented topics** ϕ'_x although both of them are represented by a multinomial distribution over items. User-oriented topics are used to model user interest, which is assumed to be generally stable over time. In contrast, time-oriented topics are used to model the temporal context (i.e., the public's attention during a particular time), which has a clear temporal feature. For example, the popularity of the topics may increase or decrease over time and reach a peak during a certain period of time, as shown in Fig. 2.2.

Definition 2.5 (*User Interest*) Given a user u, her/his intrinsic interest, denoted as θ_u, is a multinomial distribution over user-oriented topics.

Definition 2.6 (*Temporal Context*) Given a time interval t, the temporal context during t, denoted as θ'_t, is a multinomial distribution over time-oriented topics or items.

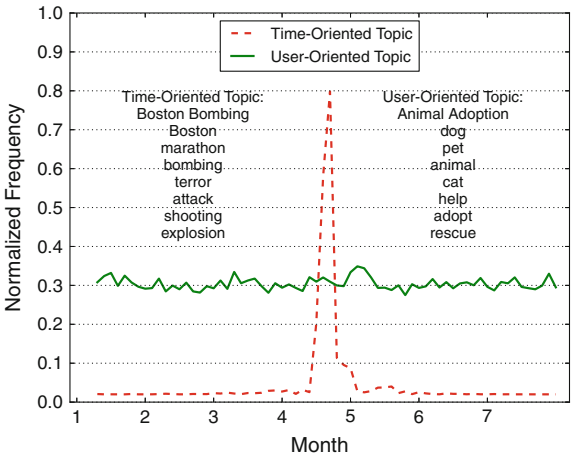

Fig. 2.2 An Example of two types of topics in delicious [22] © 2015 Association for Computing Machinery, Inc. reprinted by permission

2.2.2 Temporal Context-Aware Mixture Model

Given a rating cuboid $\boldsymbol{C}$ which stores users' rating histories, we aim to model user rating behaviors by exploiting the information captured in $\boldsymbol{C}$. Before presenting the devised model, we first describe an example to illustrate the motivation of our design.

As mentioned before, users' rating behaviors in social media systems are influenced by not only intrinsic interest but also the temporal context. It is crucial to distinguish between user-oriented topics and time-oriented topics, because the two have very different characteristics. For example, Fig. 2.2 shows an example of a user-oriented topic and a time-oriented topic detected by TCAM model from Delicious. For demonstration, we present only the top eight tags that have the highest probability under each topic. We can easily tell the difference between the two topics from both their temporal distributions and the content descriptions. For the time-oriented topic, the items (i.e., tags) are related to a certain event (e.g.,"Boston Marathon bombings"). The popularity of the topic experiences a sharp increase during a particular time interval (e.g., in April 2013). For the user-oriented topic, the items are about the user's regular interest (e.g., "Pet Adoption"). The temporal distribution of the topic does not show any spike-like fluctuation. Hence, our TCAM models the user-oriented topics and the time-oriented topics simultaneously.

To consider the influence of the user intrinsic interest and the temporal context in a unified manner, TCAM computes the likelihood that a user u will rate an item v during a time interval t as follows.

$$P(v|u,t,\Psi) = \lambda_u P(v|\theta_u) + (1-\lambda_u)P(v|\theta'_t) \tag{2.1}$$

where Ψ denotes the model parameter set, $P(v|\theta_u)$ is the probability that item v is generated from u's intrinsic interest, denoted as θ_u, and $P(v|\theta'_t)$ denotes the probability that item v is generated from the temporal context during time interval t, i.e., θ'_t. The parameter λ_u is the mixing weight which represents the influence probability of the user interest. That is, user u is influenced by personal interest θ_u with probability λ_u, and is influenced by the temporal context θ'_t with probability $1-\lambda_u$, for decision making. It is worth mentioning that TCAM holds personalized mixing weights for individual users, considering the differences between users in personalities (e.g., openness and agreeableness).

The user interest component θ_u is modeled by a multinomial distribution over user-oriented topics, and each item is generated from a user-oriented topic z. Thus, $P(v|\theta_u)$ is computed as follows.

$$P(v|\theta_u) = \sum_{z=1}^{K_1} P(v|\phi_z)P(z|\theta_u) \tag{2.2}$$

As for the temporal context component θ'_t, it is modeled as a multinomial distribution over a set of latent time-oriented topics, and each item is generated from a time-oriented topic x. Then, $P(v|\theta'_t)$ is formulated as follows:

$$P(v|\theta'_t) = \sum_{x=1}^{K_2} P(v|\phi'_x)P(x|\theta'_t). \tag{2.3}$$

As an illustrative example in Fig. 2.1, the user is influenced by personal interest and the temporal context with probabilities 0.64 and 0.36, respectively. Four user-oriented topics and three time-oriented topics are also shown, respectively, where the weights representing the user's interest distribution over the user-oriented topics as well as the temporal context distribution over the time-oriented topics are labeled in the corresponding edges. We can see that user-oriented topic U1 dominates the user's interest, and time-oriented topic T1 attracts most attentions from the general public at time t. The probabilities of topics' generating items are also labeled in the corresponding edges. For example, the weight b on the edge linking topic U1 and item v2 represents the probability of U1 generating item v2.

Figure 2.3 illustrates the generative process of TCAM with a graphical model. The structure of TCAM is similar to the PLSA model, but TCAM has additional machinery to handle the mixing weight λ_u. In particular, a latent random variable s, associated with each item, is adopted as a switch to determine whether the item is generated according to the temporal context θ'_t or the user's interest θ_u. s is sampled from a user-specific Bernoulli distribution with the mean λ_u. N indicates the number of users; K_1 is the number of user-oriented topics; K_2 is the number of time-oriented topics; T is the number of time slices and M_u is the number of items rated by u. The generative process of TCAM is summarized as follows.

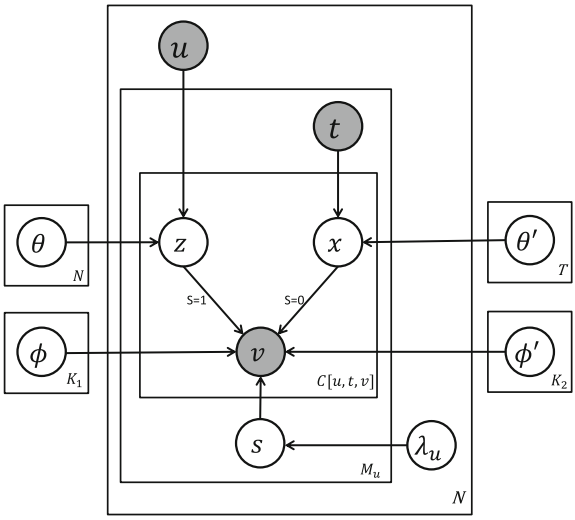

Fig. 2.3 The graphical representation of TCAM [22] © 2015 Association for Computing Machinery, Inc. reprinted by permission

For each item v rated by u at time slice t:

1. Sample s from $Bernoulli(\lambda_u)$
2. If $s = 1$
   a. Sample user-oriented topic z from $Multinomial(\theta_u)$
   b. Sample item v from $Multinomial(\phi_z)$
   c. Repeat the above two steps $C[u, t, v]$ times
3. Otherwise
   a. Sample time-oriented topic x from $Multinomial(\theta'_t)$
   b. Sample item v from $Multinomial(\phi'_x)$
   c. Repeat the above two steps $C[u, t, v]$ times

2.2.3 Model Inference

Given a rating cuboid $\boldsymbol{C}$, the learning procedure of our model is to estimate the unknown model parameter set $\Psi = \{\theta,\ \phi,\ \theta',\ \phi',\ \lambda\}$. The log-likelihood is derived as follows:

$$L(\Psi|\boldsymbol{C}) = \sum_{u=1}^{N}\sum_{t=1}^{T}\sum_{v=1}^{V} C[u, t, v] \log P(v|u, t, \Psi), \tag{2.4}$$

where $P(v|u, t, \Psi)$ is defined in Eq. (2.1).

The goal of parameter estimation is to maximize the log-likelihood in Eq. (2.4). As this equation cannot be solved directly by applying maximum likelihood estimation (MLE), we apply an EM approach instead. In the expectation (E) step of the EM approach, we introduce $P(s|u, t, v; \widehat{\Psi})$ which is the posterior probability of choosing personal interest θ_u (i.e., $s = 1$) or temporal context θ'_t (i.e., $s = 0$), respectively, given user rating behavior (u, t, v) and the current estimations of the parameters $\widehat{\Psi}$. In the maximization (M) step, parameters are updated by maximizing the expected complete data log-likelihood $Q(\Psi)$ based on the posterior probability computed in the E-step.

In the **E-step**, $P(s|u, t, v; \widehat{\Psi})$ is updated according to Bayes formulas as in Eq. (2.5).

$$P(s|u, t, v; \widehat{\Psi}) = \frac{s\lambda_u P(v|\theta_u) + (1 - s)(1 - \lambda_u)P(v|\theta'_t)}{\lambda_u P(v|\theta_u) + (1 - \lambda_u)P(v|\theta'_t)}, \tag{2.5}$$

where $P(v|\theta_u)$ and $P(v|\theta'_t)$ are defined as in Eqs. (2.2) and (2.3), respectively. To obtain the updated parameters $P(z|\theta_u)$ and $P(v|\phi_z)$, the posterior probability $P(z|s = 1, u, t, v; \widehat{\Psi})$ is computed as:

$$P(z|s = 1, u, t, v; \widehat{\Psi}) = \frac{P(v|\phi_z)P(z|\theta_u)}{\sum_{z'=1}^{K_1} P(v|\phi_{z'})P(z'|\theta_u)}. \tag{2.6}$$

Based on $P(z|s=1,u,t,v;\widehat{\Psi})$ and $P(s=1|u,t,v;\widehat{\Psi})$, we introduce the notation $P(z|u,t,v;\widehat{\Psi})$ as follows:

$$P(z|u,t,v;\widehat{\Psi}) = P(z|s=1,u,t,v;\widehat{\Psi})P(s=1|u,t,v;\widehat{\Psi}). \tag{2.7}$$

To obtain the updated parameters $P(x|\theta'_t)$ and $P(v|\phi'_x)$, we update the posterior probability $P(x|s=0,u,t,v;\widehat{\Psi})$ as follows:

$$P(x|s=0,u,t,v;\widehat{\Psi}) = \frac{P(v|\phi'_x)P(x|\theta'_t)}{\sum_{x'=1}^{K_2} P(v|\phi'_{x'})P(x'|\theta'_t)}. \tag{2.8}$$

Based on $P(x|s=0,u,t,v;\widehat{\Psi})$ and $P(s=0|u,t,v;\widehat{\Psi})$, we introduce the notation $P(x|u,t,v;\widehat{\Psi})$ as follows:

$$P(x|u,t,v;\widehat{\Psi}) = P(x|s=0,u,t,v;\widehat{\Psi})P(s=0|u,t,v;\widehat{\Psi}). \tag{2.9}$$

With simple derivations [8], we obtain the expectation of complete data log-likelihood for TCAM:

$$\begin{aligned} Q(\Psi) = \sum_{u=1}^{N}\sum_{v=1}^{V}\sum_{t=1}^{T} C[u,t,v]\{&P(s=1|u,t,v;\widehat{\Psi})\sum_{z=1}^{K_1} P(z|s=1,u,t,v;\widehat{\Psi})\log[\lambda_u P(v|\phi_z)P(z|\theta_u)] \\ &+ P(s=0|u,t,v;\widehat{\Psi})\sum_{x=1}^{K_2} P(x|s=0,u,t,v;\widehat{\Psi})\log[(1-\lambda_u)P(v|\phi'_x)P(x|\theta'_t)]\}. \end{aligned} \tag{2.10}$$

In the **M-step**, we find the estimation Ψ that maximizes the expectation of the complete data log-likelihood $Q(\Psi)$ with the constraints $\sum_{v=1}^{V} P(v|\phi_z)=1$, $\sum_{v=1}^{V} P(v|\phi'_x)=1$, $\sum_{z=1}^{K_1} P(z|\theta_u)=1$ and $\sum_{x=1}^{K_2} P(x|\theta'_t)=1$, using the following updating formulas.

$$P(z|\theta_u) = \frac{\sum_{v=1}^{V}\sum_{t=1}^{T} C[u,t,v]P(z|u,t,v;\widehat{\Psi})}{\sum_{z'=1}^{K_1}\sum_{v=1}^{V}\sum_{t=1}^{T} C[u,t,v]P(z'|u,t,v;\widehat{\Psi})} \tag{2.11}$$

$$P(v|\phi_z) = \frac{\sum_{t=1}^{T}\sum_{u=1}^{N} C[u,t,v]P(z|u,t,v;\widehat{\Psi})}{\sum_{v'=1}^{V}\sum_{t=1}^{T}\sum_{u=1}^{N} C[u,t,v']P(z|u,t,v';\widehat{\Psi})} \tag{2.12}$$

$$P(x|\theta'_t) = \frac{\sum_{v=1}^{V}\sum_{u=1}^{N} C[u,t,v]P(x|u,t,v;\widehat{\Psi})}{\sum_{x'=1}^{K_2}\sum_{v=1}^{V}\sum_{u=1}^{N} C[u,t,v]P(x'|u,t,v;\widehat{\Psi})} \tag{2.13}$$

$$P(v|\phi'_x) = \frac{\sum_{t=1}^{T}\sum_{u=1}^{N} C[u,t,v]P(x|u,t,v;\widehat{\Psi})}{\sum_{v'=1}^{V}\sum_{t=1}^{T}\sum_{u=1}^{N} C[u,t,v']P(x|u,t,v';\widehat{\Psi})} \tag{2.14}$$

With an initial random guess of Ψ , we alternately apply the E-step and M-step until a termination condition is met. To adapt to different users, we estimate the parameter λ_u in M-step, instead of picking a fixed λ value for all users. This personalized treatment can automatically adapt the model parameter estimation for various users. Specifically, λ_u is estimated as follows.

$$\lambda_u = \frac{\sum_{t=1}^{T}\sum_{v=1}^{V} C[u,t,v]P(s=1|u,t,v;\widehat{\Psi})}{\sum_{t=1}^{T}\sum_{v=1}^{V}\sum_{s=0}^{1} C[u,t,v]P(s|u,t,v;\widehat{\Psi})} \tag{2.15}$$

2.2.4 Discussion About TCAM

A number of relevant issues of the proposed TCAM model are discussed in this subsection.

Hyperparameter setting. In our model, we still have four hyperparameters to tune manually, including the number of user-oriented topics K_1, the number of time-oriented topics K_2, the number of time slices T and the number of EM iterations. K_1 and K_2 are the desired numbers of user-oriented topics and time-oriented topics, respectively, which need to be tuned empirically. T is the number of time slices used in our model to generate time-oriented topics, which provides users with the flexibility to adjust the granularity/length of the time slice. The larger T is, the more fine-grained time slices are. Regarding the number of EM iterations, we observe that convergence can be achieved in a few iterations (e.g., 50) because the model inference procedure using the EM approach is fast. The time cost of each iteration is $\mathcal{O}(NK_1V + TK_2V)$, which is very similar to the time cost required for PLSA implementation [8]. It is worth mentioning that EM algorithms can be easily expressed in MapReduce [6, 18], so the inference procedure of TCAM can be naturally decomposed for parallel processing, which is scalable to large-scale datasets.

Guidance with Dirichlet Priors. Prior knowledge can be integrated into TCAM models to guide the topic discovery process. For example, in the MovieLens dataset, we can introduce prior knowledge and guide the user-oriented topics so that they are related to the genres of movies, such as action and comedy. Another example is the Digg dataset, where we can integrate prior knowledge to guide the time-oriented topics so that they are aligned with breaking events. Specifically, we define a conjugate prior (i.e., Dirichlet prior) on each multinomial topic distribution. Let us denote the Dirichlet prior β_z for user-oriented topic z and β'_x for time-oriented topic x. $\beta_z(v)$ and $\beta'_x(v)$ can be interpreted as the corresponding pseudo counts for item v when we estimate the topic distributions $P(v|\phi_z)$ and $P(v|\phi'_x)$, respectively. With these conjugate priors, we can use the maximum a posteriori (MAP) estimator for parameter estimation, which can be computed using the same EM algorithm except that we should replace the Eqs. (2.12) and (2.14) with the following formulas, respectively:

$$P(v|\phi_z) = \frac{\sum_t \sum_u C[u,t,v] P(z|u,t,v;\widehat{\Psi}) + \beta_z(v)}{\sum_{v'} \sum_t \sum_u (C[u,t,v'] P(z|u,t,v';\widehat{\Psi}) + \beta_z(v'))},$$

$$P(v|\phi'_x) = \frac{\sum_t \sum_u C[u,t,v] P(x|u,t,v;\widehat{\Psi}) + \beta_x(v)}{\sum_{v'} \sum_t \sum_u (C[u,t,v'] P(x|u,t,v';\widehat{\Psi}) + \beta_x(v'))}.$$

Advantages of TCAM. The advantages of the TCAM model are summarized as follows. (1) We unify the influences from the user interest and the temporal context to model user rating behaviors. (2) We distinguish between user-oriented topics and time-oriented topics. Two different types of latent topics are proposed to model user interest and the temporal context, respectively. By taking away the influence of the temporal context, *user-oriented topics* can capture user intrinsic interests more precisely. Likewise, without the influence of user interests, *time-oriented topics* can better reflect the temporal context because the noise induced by a wide variety of user interests could contaminate the time-oriented topics. (3) TCAM can generate interpretable individual user profiles that can be presented alongside item recommendations to allow users to understand the rationale behind the recommendations. We will show sample user profiles in Sect. 2.4.6.

2.2.5 Item-Weighting for TCAM

In this section, we propose an item-weighting scheme to improve TCAM's performance. Similarly to traditional topic models, TCAM assumes that all items are equally important in computing generation probabilities. As a result, popular items with more ratings tend to be estimated with high generation probability and ranked in top positions in each topic, which impairs the quality of both the user-oriented topics and the time-oriented topics.

For user-oriented topics, popular items are not good indicators of user intrinsic interests. A popular item rated by many users conveys less information about a user's interest than an item rated by few users. For time-oriented topics, it is expected that items representing the public's attention at a given time should be highly ranked, such as items with bursty temporal distributions, since bursts of items are generally triggered by breaking news or events that attract the public's attention. Unfortunately, bursty items are most likely to be overwhelmed by long-standing popular items. Figure 2.4 shows the temporal frequency of the top six tags of a sample time-oriented topic discovered from Delicious. It can be observed that the topic concerns swine flu. The temporal distributions of three bursty tags, "flu," "mexico," and "swineflu", undergo sharp spikes. Although the trends of the three tags do not always synchronize, they each go through a drastic increase and reach a peak in July 2009. The bursts in these curves are triggered by a real-world event, i.e., the swine flu outbreak in Mexico. The other three tags, "news," "health," and "death", maintain high frequency throughout the year. However, they convey little information about the event.

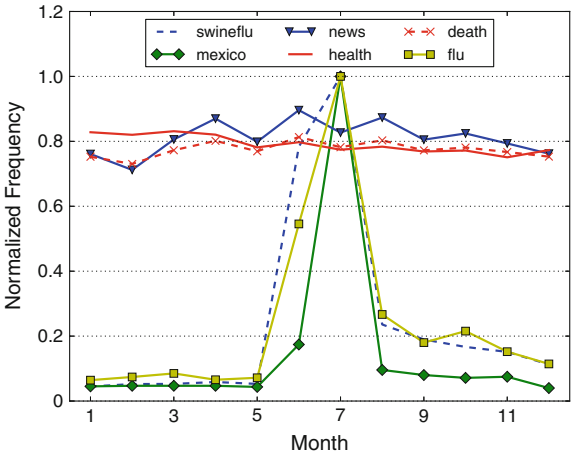

Fig. 2.4 An example of bursty tags and popular tags [22] © 2015 Association for Computing Machinery, Inc. reprinted by permission

Although they are relevant to the event, they are also related to many other topics. Hence, it is desirable to rank bursty items higher than popular item when representing time-oriented topics.

To address the challenge posed by the popular items, we propose an item-weighting scheme to reduce the importance of popular items while promoting weights for salient, but infrequent, and bursty items in computing generation probability. From the viewpoint of information theory [5], the entropy of an item v is defined as follows:

$$E(v) = -\sum_{u} P(u|v) \log P(u|v).$$

Suppose that the item v is preferred by users with equal probability $P(u|v) = \frac{1}{N(v)}$, the maximum entropy is,

$$E(v) = \log N(v).$$

Generally, the entropy of an item tends to be proportional to its frequency/popularity $N(v)$. Hence, in the following analysis, we use the maximum entropy to approximate the exact entropy to simplify the calculation.

To allow salient items to be ranked higher in use-oriented topics, the weights of items should be inversely proportional to the entropy, as discussed above. Hence, we propose a concept called *inverse user frequency* to measure the ability of items to represent salient information. Let N be the total number of users in the entire dataset; the *inverse user frequency (IUF)* for the item v is defined as follows:

$$iuf(v) = \log N - \log N(v) = \log \frac{N}{N(v)}, \tag{2.16}$$

which is similar to the inverse document frequency for a term in text mining.

To take into account the bursty information of items, we propose to compute the *bursty degree* of an item v using the following equation:

$$B(v,t) = \frac{N_t(v)}{N_t}\frac{N}{N(v)}, \tag{2.17}$$

where $N_t(v)$ represents the popularity of item v at time slice t, i.e., the number of users who rate item v at time slice t, N_t is the number of active users at time slice t, $N(v)$ is the overall popularity of v across all time slices, and N is the total number of users in the dataset.

Combining the inverse user frequency and the bursty degree of items, we assign weight to the item v as follows.

$$w(v,t) = iuf(v) \times B(v,t) \tag{2.18}$$

Integrating the weights of items defined in Eq. (2.18), we obtain the weighted user-time item cuboid $\bar{\boldsymbol{C}}$ from the original $\boldsymbol{C}$ as follows:

$$\bar{C}[u,t,v] = C[u,t,v]w(v,t). \tag{2.19}$$

It should be noted that the item-weighting scheme makes TCAM no longer correspond to a well-defined probabilistic generative process since the values stored in the cuboid $\boldsymbol{C}$ are no longer integers, but it actually improves the empirical performance of TCAM in the tasks of both temporal recommendation and topic discovery by inserting IUF and bursty weights as heuristic factors into the model inference procedures, as shown in the experiment section. From the perspective of information theory, an item with lower entropy conveys more information about a user's intrinsic interests, and the one with higher bursty degree is more capable of representing the temporal context at some specific time. The TCAM model enhanced by IUF and bursty weights incorporate these observations and intuitions.

2.3 Temporal Recommendation

The conventional top-k recommendation task can be stated as follows: given a user, the recommender system should recommend a small number, say k, of items from all the available items. Note that the conventional top-k recommendation task does not consider the temporal information. However, in reality, user rating behaviors, influenced by both user interests and the temporal context, are dynamic. For example, user u rating item v in time interval t does not mean that u still favors v in time interval $t+1$. Besides, each item has its own lifespan, especially for time-sensitive items such as news. It is undesirable to recommend outdated news. Hence, an ideal recommender system is expected to have the ability to recommend the right item v to user u in the right time interval t, rather than in other time intervals. In this chapter, we propose

the task of temporal top-k recommendation as follows: given a query $q = (u,\ t)$, i.e., a querying user u with a time interval t, the recommender model recommends k items which match u's interests and the temporal context at t.

Below, we will present how to deploy TCAM to facilitate temporal recommendations. Once we have inferred model parameters of TCAM, such as user interest θ, temporal context θ', user-oriented topics ϕ, time-oriented topics ϕ' and mixing weights λ, given a query $q = (u,\ t)$, a ranking score $S(u, t, v)$ for each item v can be computed according to Eq. (2.21), and then the top-k items with highest ranking score will be returned. Specifically, when receiving a query $q = (u,\ t)$, a new multinomial distribution for the query, ϑ_q, is first constructed by combining θ_u and ${\theta'}_t$. More specifically, we expand the user interest and temporal context spaces to be of the same dimension. For example, if there are K_1 user-oriented topics and K_2 time-oriented topics, the expanded topic space will have $K = K_1 + K_2$ topics. The expanded user interest distribution is defined as $\widetilde{\theta}_u =< \theta_u, 0, \cdots, 0 >$, where we set 0 on the time-oriented topics. Similarly, we define the expanded temporal context distribution to be $\widetilde{\theta}'_t =< 0, \ldots, 0, \theta'_t >$. The new distribution is defined as $\vartheta_q = \lambda_u \widetilde{\theta}_u + (1 - \lambda_u)\widetilde{\theta}'_t$. Correspondingly, we renumber the time-oriented topic x and change its range from $[1, \ldots, K_2]$ to $[K_1 + 1, \ldots, K]$. Then, we use $\varphi_{\widetilde{z},v}$ to denote the weight of item v on dimension $\widetilde{z}$ that corresponds to user-oriented topic z or time-oriented topic x, which depends on the value of $\widetilde{z}$, as shown in Eq. (2.20).

$$\varphi_{\widetilde{z},v} = \begin{cases} \phi_{z,v} & \widetilde{z} \leq K1 \\ {\phi'}_{x,v} & \widetilde{z} > K1 \end{cases} \tag{2.20}$$

$$S(u, t, v) = \sum_{\widetilde{z}=1}^{K} \vartheta_{q,\widetilde{z}} \varphi_{\widetilde{z},v} \tag{2.21}$$

2.4 Experiments

In this section, we experimentally evaluate the performance of our proposed models.

2.4.1 Datasets

Our experiments are conducted on four real datasets: Digg, MovieLens, Douban Movie, and Delicious. The basic statistics of the four datasets are shown in Table 2.2. Only the implicit feedback data can be available in Digg and Delicious datasets, so we compute the cell value $C[u, t, v]$ for these two datasets according to the frequency/number of the interaction between user u and item v at time t. For Douban

Table 2.2 Basic statistics of the four datasets

	Digg	MovieLens	Douban movie	Delicious
# of users	139,409	71,567	33,561	201,663
# of items	3,553	10,681	87,081	2,828,304
# of ratings	3,018,197	10,000,054	5,257,665	36,966,661
time span(year)	2009–2010	1998–2009	2005–2010	2008–2009

Movie and MovieLens datasets, the explicit feedback information is available. Hence, the user-time item rating cuboid $\boldsymbol{C}$ can be directly derived from users' star ratings.

- **Digg**. Digg is a popular social news aggregator, which allows users to vote news stories up or down, called *digging* or *burying*, respectively. The Digg dataset used in our experiment is Digg2009 [9], a publicly available dataset containing 3,018,197 votes on 3553 popular stories cast by 139,409 distinct users. This dataset also records the friendship network of these Digg users. Although this dataset contains only the IDs of news stories (the titles and the contents of stories are excluded), it is sufficient to evaluate the effectiveness of user behavior modeling in our work.
- **Douban Movie**. Douban[2] is the largest movie review website in China. In total, we crawled 33,561 unique users and 87,081 unique movies with 5,257,665 movie ratings.
- **MovieLens**. MovieLens is a publicly available movie dataset from the web-based recommender system MovieLens. The dataset contains 10M ratings on a scale from 1 to 5 made by 71567 users on 10681 movies. We selected users who had rated at least 20 movies.
- **Delicious**. Delicious is a collaborative tagging system where users can upload and tag web pages. We collected 201,663 users and their tagging behaviors during the period Feb. 2008–Dec. 2009. The dataset contains 2,828,304 tags. Topics on technology and electronics account for about half of all web pages. Most of the other web pages are about breaking news with strong temporal features.

Note that the Douban Movie and Delicious datasets are collected by ourselves, and we make them publicly available.[3]

2.4.2 Comparisons

The temporal context-aware mixture model (TCAM) was outlined in Sect. 2.2. TCAM can be enhanced by the item-weighting scheme, which leads to a weighted TCAM, called WTCAM. We compare them with four categories of competitor approaches.

[2] http://douban.com.

[3] http://net.pku.edu.cn/daim/hongzhi.yin/.

- **User-Topic Model (UT)**. We implemented a user-topic model following the previous works [12, 15]. This model is similar to the classic author-topic model (AT model) [14] which assumes that topics are generated according to user interests. A user document D_u is regarded as a sample of the following mixture model:

$$P(v|u;\Psi) = \lambda_B P(v|\theta_B) + (1-\lambda_B)\sum_z P(z|\theta_u)P(v|\phi_z),$$

  where v is an item (or word) in user document D_u, $P(z|\theta_u)$ is the probability of user u choosing the zth topic ϕ_z. θ_B is a background model and λ_B is the mixing weight for it. The purpose of using a background model θ_B is to make the topics learned from the dataset more discriminative; since θ_B gives high probabilities to nondiscriminative and noninformative items or words, we expect such items or words to be accounted for by θ_B and thus the topic models to be more discriminative. In a nutshell, the background model θ_B is used to capture common items or words.
- **Time-Topic Model (TT)**. Following previous works [11, 16], we implemented a time-topic model. This model considers only the temporal information and ignores user interests. TT assumes that topics are generated by the temporal context, and that user behaviors are influenced by the temporal context. The probabilistic formula of the time-topic model is presented as follows:

$$P(v|t;\Psi) = \lambda_B P(v|\theta_B) + (1-\lambda_B)\sum_x P(x|\theta'_t)P(v|\phi'_x),$$

  where $P(x|\theta'_t)$ is the probability of the general public choosing xth topic during time period t, and θ_B is a background model that plays the same role with the one in the above UT model.
- **BPRMF**. This is a state-of-the-art matrix factorization model for item ranking that is optimized using BPR [13]. This model outperforms most of the existing recommender models in the task of top-k item recommendation. We used the BPRMF implementation provided by MyMediaLite, a free software recommender system library [7].
- **BPTF**. This is a state-of-the-art recommender model for rating prediction that uses a probabilistic tensor factorization technique by introducing additional factors for time [19]. This model outperforms most of the existing recommender models that consider time information.

2.4.3 Evaluation Methodology

To make the evaluation process fair and reproducible, we adopt the *methodological description framework* proposed in [2] 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*?
5. Which items are considered as *relevant items* for each user?

Base set condition. The base set conditions state whether the splitting procedure of training and test sets is based on the whole set of ratings $\boldsymbol{C}$, or on each of the sub-datasets of $\boldsymbol{C}$ independently. We adopt the *user-centered base set condition* where we perform the splitting independently on each user's ratings, ensuring that all users will have ratings in both the training and test sets.

Rating order and size conditions. We adopt the *time-dependent rating order condition*. Specifically, for each user u, his/her ratings $S(u)$ are ranked according to their rated timestamps. We use the 80-th percentile as the cut-off point so that ratings before this point will be used for training and the rest are for testing, i.e., $S(u)$ is divided into the training set $S^{train}(u)$ and test set $S^{test}(u)$.

Target item condition. To simulate a real-world setting, we require each tested recommender system to rank all the items except the target user's training items. In other words, given a target user u, each tested recommender system has to find top-k items from all available items except those in the set $S^{train}(u)$.

Relevant item condition. Relevant item conditions select the items to be interpreted as relevant for the target user. The notion of relevance is central for information retrieval metrics applied to evaluate top-k recommendations. We adopt the *test-based relevant items* condition in which the set of relevant items for target user u is formed by the items in u's test set $S^{test}(u)$.

The above condition combination is also called "uc_td_prop" for short. We repeat our experiments on four different social media datasets for increasing the generalization of the evaluation results, and we use a hold-out procedure on each dataset.

To make the experimental results comparable and reproducible, we use multiple well-known metrics to measure the ranked results. Similar to evaluations in information retrieval, we first use Precision@k to assess the quality of the top-k recommended items as follows:

$$Precision@k = \frac{\#relavances}{k},$$

where #*relavances* is the number of relevant items in the top-k recommended items. We also consider NDCG, a widely used metric for a ranked list. NDCG@k is defined as:

$$NDCG@k = \frac{1}{IDCG} \times \sum_{i=1}^{k} \frac{2^{r_i} - 1}{\log(i+1)},$$

where r_i is 1 if the item at position i is a "relevant" item and 0 otherwise. IDCG is chosen for the purpose of normalization so that the perfect ranking has an NDCG value of 1. Considering that some users may have a large number of items in the test data while some have just a few, we also adopt the $F1$ score as our metric.

2.4.4 Recommendation Effectiveness

Tables 2.3, 2.4, 2.5 and 2.6 report the performance of the proposed models and other competitors in terms of Precision@k, NDCG@k and F1@k on Digg, MovieLens, Douban Movie and Delicious datasets, respectively. From the reported results, we observe that:

1. Our proposed models TCAM and W-TCAM consistently outperform other competitors such as UT, TT, BPRMF and BPTF on all four datasets. This observation shows that recommendation accuracy, especially temporal recommendation accu-

Table 2.3 Temporal recommendation accuracy on digg dataset

Methods	Precision			NDCG			F1 Score		
	P@1	P@5	P@10	N@1	N@5	N@10	F1@1	F1@5	F1@10
UT	0.091	0.086	0.084	0.093	0.091	0.088	0.007	0.028	0.044
TT	0.182	0.149	0.126	0.178	0.148	0.139	0.017	0.041	0.071
BPRMF	0.048	0.045	0.040	0.048	0.040	0.037	0.002	0.015	0.022
BPTF	0.194	0.166	0.152	0.195	0.176	0.165	0.017	0.050	0.080
TCAM	0.237	0.210	0.179	0.234	0.203	0.188	0.017	0.056	0.093
W-TCAM	**0.258**	**0.220**	**0.201**	**0.252**	**0.224**	**0.208**	**0.019**	**0.063**	**0.098**

Table 2.4 Temporal Recommendation Accuracy on MovieLens Dataset

Methods	Precision			NDCG			F1 Score		
	P@1	P@5	P@10	N@1	N@5	N@10	F1@1	F1@5	F1@10
UT	0.343	0.252	0.211	0.338	0.257	0.234	0.036	0.099	0.136
TT	0.260	0.193	0.168	0.248	0.198	0.186	0.024	0.070	0.093
BPRMF	0.342	0.239	0.192	0.303	0.229	0.195	0.034	0.084	0.119
BPTF	0.383	0.270	0.224	0.365	0.290	0.261	0.035	0.103	0.141
TCAM	0.385	0.304	0.259	0.406	0.325	0.286	0.037	0.115	0.155
W-TCAM	**0.401**	**0.324**	**0.264**	**0.427**	**0.350**	**0.313**	**0.040**	**0.123**	**0.165**

Table 2.5 Temporal Recommendation Accuracy on Douban Movie Dataset

Methods	Precision			NDCG			F1 Score		
	P@1	P@5	P@10	N@1	N@5	N@10	F1@1	F1@5	F1@10
UT	0.141	0.104	0.087	0.139	0.106	0.097	0.015	0.041	0.056
TT	0.105	0.078	0.068	0.101	0.080	0.075	0.010	0.028	0.038
BPRMF	0.138	0.101	0.085	0.136	0.103	0.094	0.015	0.040	0.055
BPTF	0.158	0.111	0.092	0.151	0.119	0.108	0.015	0.043	0.058
TCAM	0.168	0.133	0.113	0.177	0.142	0.125	0.016	0.050	0.068
W-TCAM	**0.175**	**0.141**	**0.115**	**0.186**	**0.153**	**0.137**	**0.018**	**0.054**	**0.072**

Table 2.6 Temporal Recommendation Accuracy on Delicious Dataset

Methods	Precision			NDCG			F1 Score		
	P@1	P@5	P@10	N@1	N@5	N@10	F1@1	F1@5	F1@10
UT	0.086	0.076	0.073	0.082	0.081	0.078	0.006	0.025	0.039
TT	0.104	0.085	0.072	0.102	0.085	0.080	0.010	0.024	0.041
BPRMF	0.065	0.059	0.051	0.064	0.062	0.060	0.005	0.019	0.030
BPTF	0.111	0.095	0.087	0.112	0.101	0.095	0.010	0.029	0.046
TCAM	0.136	0.120	0.103	0.134	0.116	0.108	0.010	0.032	0.053
W-TCAM	**0.148**	**0.126**	**0.113**	**0.144**	**0.128**	**0.119**	**0.011**	**0.036**	**0.056**

racy, can be improved by simultaneously considering both user intrinsic interests and the temporal context.

2. BPTF performs better than other competitor methods such as BPRMF, UT and TT because it also exploits the temporal context information when recommending items, but our proposed TCAM and W-TCAM consistently outperform BPTF. This may be because BPTF is designed for rating prediction rather than the top-k recommendation. It relies on high quality explicit feedback data (e.g., users' explicit star rating for items), however, which is not always available [13]. In contrast, our proposed TCAM and W-TCAM are suitable for both explicit and implicit user feedback data.
3. W-TCAM achieves higher temporal recommendation accuracy than TCAM, which demonstrates the benefits gained by the item-weighting scheme.
4. Comparing UT and TT, we find that UT performs better than TT on the MovieLens and Douban Movie datasets, while TT beats UT on the Digg dataset. This may be because news is a type of time-sensitive item while movies are not so time-sensitive and have longer life-span.

2.4.5 Temporal Context Influence Study

This section studies the influence degrees of users' personal interests and the temporal context on users' decision making. The user interest influence probability λ_u and the temporal context influence probability $1 - \lambda_u$ are learnt automatically in the TCAM model. We are interested in how significantly the temporal context influences the user's decisions on different social media platforms.

Since different people have different mixing weights, we plot the distributions of both the personal interest and the temporal context influence probabilities across all users. The results on the MovieLens data set are shown in Fig. 2.5, where Fig. 2.5a plots the cumulative distribution of personal interest influence probabilities, and Fig. 2.5b shows the temporal context influence probabilities. It is observed that, in general, people's personal interest influence is significantly higher than the influence

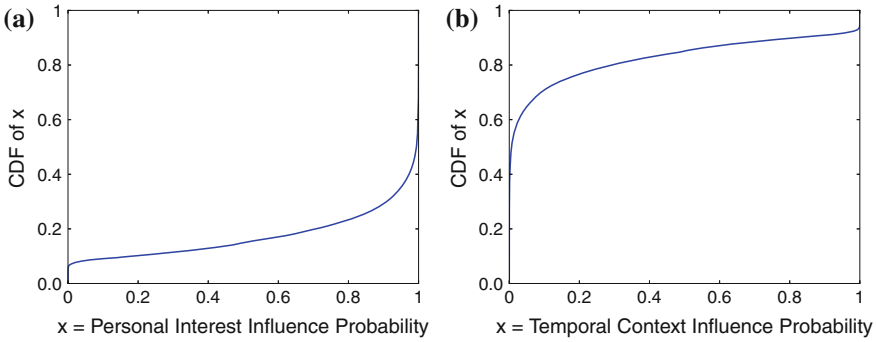

Fig. 2.5 Temporal context influence result (MovieLens). **a** Personal interest influence. **b** Temporal context influence

of the temporal context. For example, Fig. 2.5a shows that the personal interest influence probability of more than 76 % of users is higher than the 0.82. This observation indicates that most movies consumed by users are selected in accordance with their interests and tastes.

Figure 2.6a, b show, respectively, the personal interest influence probabilities and temporal context influence probabilities learnt from the Digg data. As shown in Fig. 2.6a, the personal interest influence probability is smaller than the temporal context influence probability. For example, the temporal context influence probability of more than 70 % of users is higher than 0.5. The implication of this finding is that people are mainly influenced by the temporal context when choosing news to read. By comparing the analysis results obtained from the two datasets, we observe that the temporal context influence on users' choice of news to read is much more significant than it is on the selection of movies to watch. This is probably because news is a time-sensitive item that is driven by offline social events, while movies are not so time-sensitive.

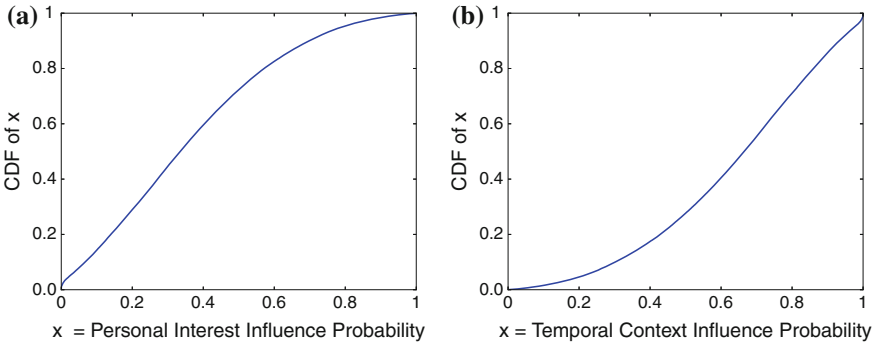

Fig. 2.6 Temporal context influence result (Digg). **a** Personal interest influence. **b** Temporal context influence

2.4.6 User Profile Analysis

Both the user interests and temporal context, as well as their influences on users' decision making, can be learnt by our TCAM model to build user profiles. This section first analyzes two sample user profiles to enable a better interpretation of user rating behaviors. Figure 2.7 shows the profiles of user 102 and user 384 learnt by TCAM from the Digg dataset. As shown in the figure, users 102 and 384 are influenced by the temporal context with influence probability values 0.88 and 0.76, respectively. We also show top-4 user-oriented topics with highest probabilities in θ_u. The weights on the edges indicate users' preferences for the topics. Note that we only choose top-4 user-oriented topics for demonstration, thus the sum of the weights on the edges is not equal to 1. There is only one overlapping user-oriented topic for users 102 and 384, and the dominating user-oriented topics for them are different (i.e., $U1$ vs. $U8$).

We also show two sample temporal context profiles for time slices 6 and 7 in Fig. 2.8. We present the top-4 time-oriented topics with highest probabilities in each θ'_t. The weights on the edges indicate the preference degrees of the general public for the chosen four time-oriented topics. We choose two temporal context profiles learnt by TCAM from Digg dataset for demonstration. By comparing the two adjacent temporal context profiles, we observe that the general public's preferences for time-oriented topics evolve over time. While the two adjacent time slices share the time-oriented topics $T1$ and $T16$, the general public focus more on topic $T1$ at time slice 6 and show more interest in topic $T6$ at time slice 7.

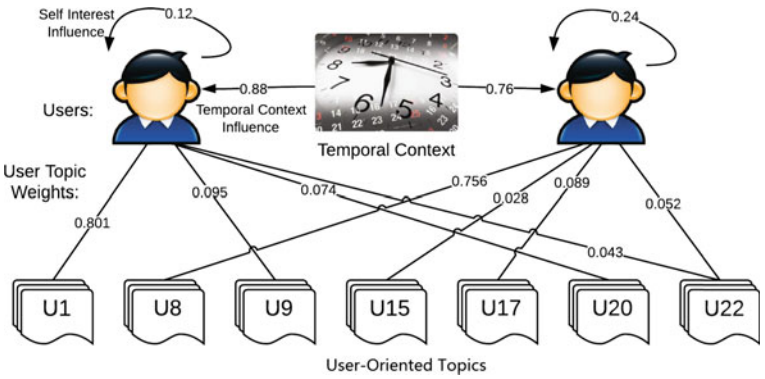

Fig. 2.7 Sample user profiles and temporal context influence learnt by TCAM

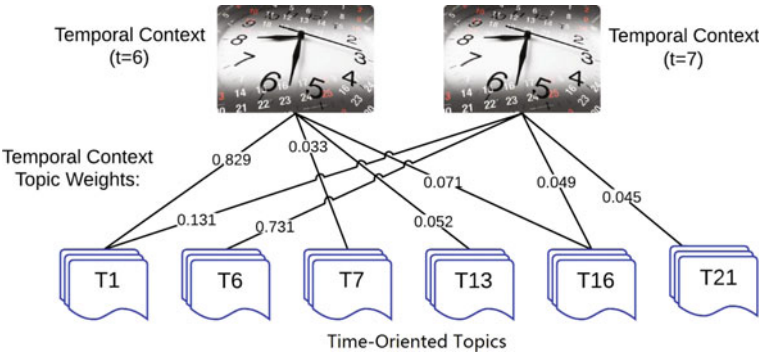

Fig. 2.8 Sample temporal context profiles

2.5 Summary

In this chapter, we focused on the problem of user behavior modeling in social media systems and its applications in temporal recommendation. Based on the intuition and observations that users' rating behaviors are influenced by two factors: user intrinsic interests as an internal factor, and the temporal context (i.e., the public's attention during a time period) as an external factor, we proposed a temporal context-aware mixture model (TCAM) that explicitly introduces two types of latent topics to model user interests and temporal context, respectively. An item-weighting scheme was developed to enhance the TCAM models by exploiting the frequency distribution and temporal distribution information of items. To demonstrate the applicability of TCAM, we deployed this model to facilitate temporal recommendation. We conducted extensive experiments on four large-scale real social media datasets, and the results and analysis demonstrated the superiority of our TCAM model over existing methods in the task of temporal recommendation, which verified our motivation. We also explored other applications for our TCAM model beyond time-aware recommendation, such as user profiling and temporal context modeling in an illustrative way in the experiment section.

References

1. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. In: Journal of MachineLearning Research, pp. 993–1022 (2003)
2. Campos, P.G., Díez, F., Cantador, I.: Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. User Model. User-Adapt. Interact. **24**(1–2), 67–119 (2014)
3. Cha, Y., Bi, B., Hsieh, C.-C., Cho, J.: Incorporating popularity in topic models for social network analysis. In: SIGIR, pp. 223–232 (2013)
4. Cha, Y., Cho, J.: Social-network analysis using topic models. In: SIGIR, pp. 565–574 (2012)

5. Cover, T.M., Thomas, J.A.: Elements of Information Theory. Wiley-Interscience, New Jersey (1991)
6. Das, A.S., Datar, M., Garg, A., Rajaram, S.: Google news personalization: scalable online collaborative filtering. In: WWW, pp. 271–280 (2007)
7. Gantner, Z., Rendle, S., Freudenthaler, C., Schmidt-Thieme, L.: Mymedialite: a free recommender system library. In: RecSys, pp. 305–308 (2011)
8. Hofmann, T.: Probabilistic latent semantic analysis. In: UAI (1999)
9. Lerman, K., Ghosh, R.: Information contagion: an empirical study of the spread of news on digg and twitter social networks. In: ICWSM (2010)
10. Liu, J., Dolan, P., Pedersen, E.R.: Personalized news recommendation based on click behavior. In: IUI, pp. 31–40 (2010)
11. Mei, Q., Zhai, C.: Discovering evolutionary theme patterns from text: an exploration of temporal text mining. In: KDD, pp. 198–207 (2005)
12. Michelson, M., Macskassy, S.A.: Discovering users' topics of interest on twitter: a first look. In: AND, pp. 73–80 (2010)
13. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: Bpr: Bayesian personalized ranking from implicit feedback. In: UAI, pp. 452–461 (2009)
14. Rosen Zvi, M., Griffiths, T., Steyvers, M., Smyth P.: The author-topic model for authors and documents. In: UAI, pp. 487–494 (2004)
15. Stoyanovich, J., Amer Yahia, S., Marlow, C., Yu, C.: Leveraging tagging to model user interests in del.icio.us. In: AAAI (2008)
16. Wang, X., Zhai, C., Hu, X., Sproat, R.: Mining correlated bursty topic patterns from coordinated text streams. In: KDD, pp. 784–793 (2007)
17. Wen, Z., Lin, C.Y.: On the quality of inferring interests from social neighbors. In: KDD, pp. 373–382 (2010)
18. Wolfe, J., Haghighi, A., Klein, D.: Fully distributed em for very large datasets. In: ICML, pp. 1184–1191 (2008)
19. Xiong, L., Chen, X., Huang, T.-K., Schneider, J.G., Carbonell, J.G.: Temporal collaborative filtering with bayesian probabilistic tensor factorization. In: SDM, pp. 211–222 (2010)
20. Xu, Z., Zhang, Y., Wu, Y., Yang, Q.: Modeling user posting behavior on social media. In: SIGIR, pp. 545–554 (2012)
21. Yin, H., Cui, B., Chen, L., Hu, Z., Huang, Z.:A temporal context-aware model for user behavior modeling in social media systems. In: SIGMOD, pp. 1543–1554 (2014)
22. Yin, H., Cui, B., Chen, L., Hu, Z., Zhou, X.: Dynamic user modeling in social media systems. ACM Trans. Inf. Syst. **33**(3), 10:1–10:44 (2015)
23. Yin, H., Cui, B., Li, J., Yao, J., Chen, C.: Challenging the long tail recommendation. Proc. VLDB Endow. **5**(9), 896–907 (2012)

Chapter 3
Spatial Context-Aware Recommendation

Abstract As a user can only visit a limited number of venues/events and most of them are within a limited distance range, the user-item matrix is very sparse, which creates a big challenge for traditional collaborative filtering-based recommender systems. The problem becomes more challenging when people travel to a new city where they have no activity history. In this chapter, we propose LCARS, a location-content-aware recommender system that offers a particular user a set of venues (e.g., restaurants) or events (e.g., concerts and exhibitions) by giving consideration to both personal interest and local preference. This recommender system can facilitate people's travel not only near the area in which they live, but also in a city that is new to them. We evaluate the performance of our recommender system on two large-scale real datasets, DoubanEvent, and Foursquare. The results show the superiority of LCARS in recommending spatial items for users, especially when traveling to new cities.

Keywords Location-based service · New city recommendation · Transferring user interest · Local preference

3.1 Introduction

Newly emerging event-based social network services (EBSNs), such as Meetup (www.meetup.com), Plancast (www.plancast.com) and DoubanEvent (www.douban.com/events/) have provided convenient online platforms for users to create, spread, track, and attend social *events* which are going to be held in some physical locations [12]. On these web services, users may propose social events, ranging from informal get togethers (e.g., movie night and dining out) to formal activities (e.g., culture salons and business meetings) by specifying when, where and what the event is. After the created event is available to the public, other users may express their intent to join event by replying "yes," "no," or "maybe" online. Meanwhile, the advances in location-acquisition and wireless communication technologies enable users to add a location dimension to traditional networks, fostering a growth of location-based social networking services (LBSNs), such as Foursquare (https://foursquare.com)

H. Yin and B. Cui, *Spatio-Temporal Recommendation in Social Media*,
SpringerBriefs in Computer Science, DOI 10.1007/978-981-10-0748-4_3

and Gowalla (www.gowalla.com) which allow users to "check-in" at spatial *venues* (e.g., restaurants in New York) via mobile devices.

In this chapter, we aim to mine more knowledge from the user activity history data in LBSNs and EBSNs to answer two typical types of questions that we often ask in our daily: (1) If we want to visit venues in a city such as Beijing, where should we go? (2) If we want to attend local events such as dramas and exhibitions in a city, which events should we attend? In general, the first question corresponds to venue recommendations, and the second question corresponds to event recommendations. By answering these two questions, we can satisfy the personalized information needs for many users in their daily routines and trip planning. For simplicity, we propose the notion of *spatial items* to denote both venues and events in a unified way, so that we can define our problem as follows: given a querying user u with a querying city l_u, find k interesting spatial items within l_u, that match the preference of u.

However, inferring user preferences for spatial items is very challenging using users' activity history in an EBSN or LBSN. First, a user can only visit a limited number of physical venues and attend a limited number of social events. This leads to a sparse user-item matrix for most existing location-based recommender systems [7, 10], which directly use collaborative filtering-based methods [14] over spatial items. Second, the observation of travel locality [10] makes the task more challenging considering that a user travels to a new place where he/she does not have any activity history. The observation of travel locality made on EBSNs and LBSNs shows that users tend to travel a limited distance when visiting venues and attending events. An investigation shows that the activity records generated by users in their non-home cities are very few and only take up 0.47 % of the activity records they left in their home cities. This observation of travel locality is quite common in the real world [16], aggravating the data sparsity problem with personalized spatial item recommendations (e.g., if we want to suggest spatial items located in Los Angeles to people from New York City). In this case, solely using a CF-based method is not feasible any more, especially when coping with the *new city* problem, because a querying user usually does not have enough activity history of spatial items in a city that is new to him/her.

Let us assume, for example, that querying user u is a Shopaholic and often visits shopping mall v' in his/her home city; v is a popular local shopping mall in city l_v that is new to u. Intuitively, a good recommender system should recommend v to u when he/she travels to l_v. However, the pure CF-based methods fail to do so. For the item-based CF [11, 15], there are few common users between v and v' according to the property of travel locality, resulting in the low similarity between the two items' user vectors. For the user-based CF [1], it is most likely that all the k nearest neighbors of user u live in the same city as u, and that few of them have visited v according to the property of travel locality.

To this end, we proposed a location-content-aware recommender system (LCARS) in [21, 22] that exploits both the location and content information of spatial items to alleviate the data sparsity problem, especially the *new city* problem. As is shown in Fig. 3.1, LCARS consists of two main parts: offline modeling and online recommendation. The offline model, LCA–LDA, is designed to model user preferences to

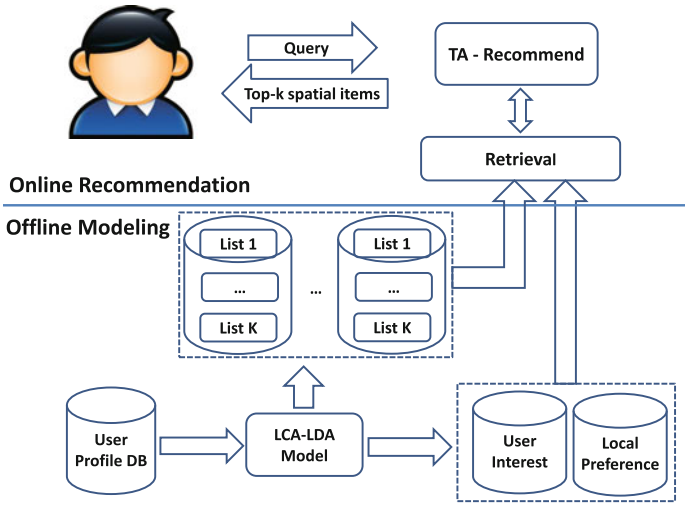

Fig. 3.1 The architecture framework of LCARS [21] © 2014 Association for Computing Machinery, Inc. Reprinted by permission

spatial items by simultaneously considering the following two factors in a unified manner. (1) *User Interest*: Music lovers may be more interested in concerts while Shopaholics would pay more attention to shopping malls. (2) *Local Preference*: When users visit a city, especially a city that is new to them, they are more likely to see local attractions and attend events that are popular in the city. Thus, the preferences of local people are a valuable resource for making a recommendation, especially when people travel to an unfamiliar area where they have little knowledge about the neighborhood. LCA–LDA can automatically learn both user interest and local preference from the user activity history. Exploiting local preference can address the issue of data sparsity to some extent. To further alleviate the data sparsity problem, LCA–LDA exploits the content information (e.g., item tags or category words) of spatial items to link content-similar spatial items together, facilitating people's travel not only near their home regions but also to cities that are new to them. It is worth mentioning that LCA–LDA can also capture the item cooccurrence patterns to link relevant items together, just like item-based collaborative filtering methods. To our best knowledge, ideas for unifying the influence of local preferences, collaborative filtering and content-based recommendation are unexplored and very challenging.

Given a querying user u with a querying city l_u, the online recommendation part computes a ranking score for each spatial item v within l_u by automatically combining u's interest and the local preference of l_u, and select top-k ones with highest ranking score as recommendation results. To speed up the process of online recommendation, we propose a scalable query processing technique for top-k recommendations which separates the offline scoring computation from online scoring computation to minimize the query time. Specifically, we partition all spatial items into regions at a given level such as cities. For each region, as is shown in Fig. 3.1, we store K lists

of spatial items that fall into the location and each list is sorted by the items' offline score in the corresponding dimension z. At query time, we retrieve all spatial items within l_u by running fast online recommendation algorithms (e.g., TA) to compute the top-k spatial items by combining the score of each candidate item from K scorers.

The remainder of the chapter is organized as follows. Section 3.2 details the location-content-aware recommender system LCARS. We report the experimental results in Sect. 3.3 and conclude the chapter in Sect. 3.4.

3.2 Location-Content-Aware Recommender System

In this section, we first introduce the key data structures and notations used in this chapter, and then present the offline modeling part and online recommendation part of our proposed location-content-aware recommender system.

3.2.1 Preliminary

For ease of the following presentation, we define the key data structures and notations used in this chapter. Table 3.1 lists the relevant notations used in this chapter.

Definition 3.1 (*Spatial Item*) A spatial item v refers to either an event or venue generated in various EBSNs or LBSNs.

Definition 3.2 (*User Activity*) A user activity is a triple (u, v, l_v) that means user u selects a spatial item v in geographical region l_v. Information about the user activity history is given by $S \subseteq \mathscr{U} \times \mathscr{V} \times \mathscr{L}$, where user activities are positive observations in the past.

The dataset D used for our model learning consists of four elements, and they are users, spatial items, geographical regions and content words, i.e., $(u, v, l_v, c_v) \in D$ where $u \in \mathscr{U}$, $v \in \mathscr{V}$, $l_v \in \mathscr{L}$, and $c_v \in \mathscr{C}_v$ (i.e., $\mathscr{C}_v$ denotes the content word set associated with spatial item v). Note that a spatial item may contain multiple content words. For an activity history record of a user u selecting a spatial item v located in l_v, we have a set of four-tuples, i.e., $D_{uv} = \{(u, v, l_v, c_v) : c_v \in \mathscr{C}_v\}$.

Definition 3.3 (*User Profile*) For each user u in the dataset D, we create a user profile D_u, which is a set of four-tuples (i.e., (u, v, l_v, c_v)) associated with u. Clearly, $D_{uv} \subseteq D_u$.

Definition 3.4 (*Topic*) A topic z in a spatial item collection $\mathscr{V}$ is represented by a topic model ϕ_z, which is a probability distribution over spatial items, i.e., $\{P(v|\phi_z) : v \in \mathscr{V}\}$ or $\{\phi_{zv} : v \in \mathscr{V}\}$. By analogy, a topic in a content word collection $\mathscr{C}$ is represented by a topic model ϕ'_z, which is a probability distribution over content words, i.e., $\{P(c|\phi'_z) : c \in \mathscr{C}\}$ or $\{\phi'_{zc} : c \in \mathscr{C}\}$.

Table 3.1 Notations used in the chapter

Symbol	Description
N, V, M, C	The number of users, spatial items, regions, content words
$\mathscr{U}, \mathscr{V}, \mathscr{L}, \mathscr{C}$	The set of users, spatial items, regions, content words
$\mathscr{V}_l$	The set of spatial items located in region l
K	The number of topics
D_u	The profile of user u
v_{ui}	The spatial item of ith record in user profile D_u
θ_u	The interest of user u, expressed by a multinomial distribution over topics
θ'_l	The local preference of region l, expressed by a multinomial distribution over topics
ϕ_z	A multinomial distribution over spatial items specific to topic z
ϕ'_z	A multinomial distribution over content words specific to topic z
z_{ui}	The topic assigned to spatial item v_{ui}
l_{ui}	The region of spatial item v_{ui}
l_u	The querying region of the querying user u
c_{ui}	A content word describing spatial item v_{ui}
$\mathscr{C}_{ui}$	The set of content words describing spatial item v_{ui}
s_{ui}	If spatial item v_{ui} is generated by θ_u or $\theta'_{l_{ui}}$
β, β'	Dirichlet priors to multinomial distributions ϕ_z, ϕ'_z
α, α'	Dirichlet priors to multinomial distributions θ_u, θ'_l
λ_u	The mixing weight specific to user u
γ, γ'	Beta priors to generate λ_u

It is worth mentioning that each topic z corresponds to two topic models in our work, i.e., ϕ_z and ϕ'_z. This design enables ϕ_z and ϕ'_z to be mutually influenced and enhanced during the topic discovery process, facilitating the clustering of content-similar spatial items into the same topic with high probability.

Definition 3.5 (*User Interest*) The intrinsic interest of user u is represented by θ_u, a probability distribution over topics.

Definition 3.6 (*Local Preference*) The local preference of geographical region l is represented by θ'_l, a probability distribution over topics. This modeling method can capture local folk-customs.

3.2.2 Model Description

In this subsection, we first describe the offline modeling part of LCARS, a probabilistic generative model called LCA–LDA.

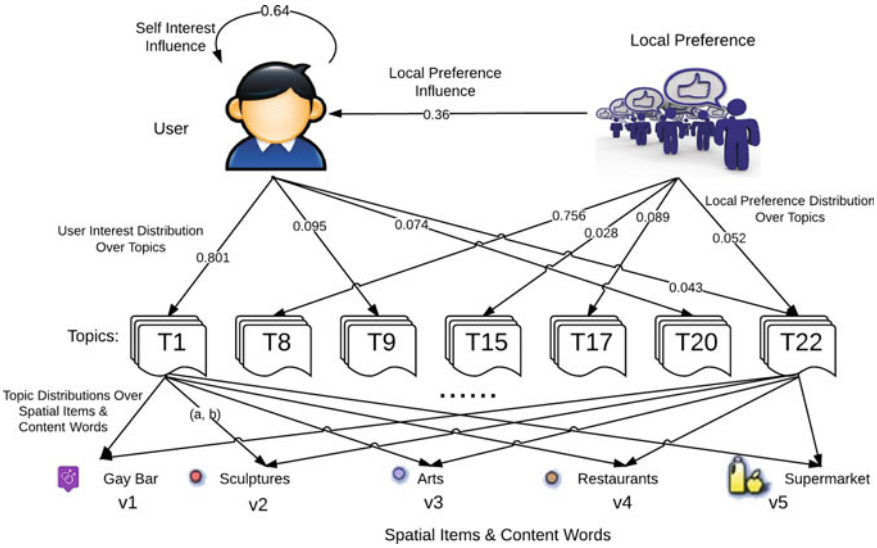

Fig. 3.2 Example of LCA–LDA Model [21] © 2014 Association for Computing Machinery, Inc. Reprinted by permission

The proposed offline modeling part, LCA–LDA, is a location-content-aware probabilistic mixture generative model that aims to mimic the process of human decision making on spatial items. As shown in Fig. 3.2, LCA–LDA considers both user's personal interest and the influence of local preference in a unified manner, and automatically leverages the effect of the two factors. Specifically, given a querying user u with a querying city l_u, the likelihood that user u will prefer item v when traveling to city l_u, is computed according to the following LCA–LDA model.

$$P(v|\theta_u, \theta'_{l_u}, \phi, \phi') = \lambda_u P(v|\theta_u, \phi, \phi') + (1 - \lambda_u) P(v|\theta'_{l_u}, \phi, \phi') \tag{3.1}$$

where $P(v|\theta_u, \phi, \phi')$ is the probability that spatial item v is generated according to the personal interest of user u, denoted as θ_u, and $P(v|\theta'_{l_u}, \phi, \phi')$ denotes the probability that spatial item v is generated according to the local preference of l_u, denoted as θ'_{l_u}. The parameter λ_u is the mixing weight which controls the motivation choice. That is, when deciding individual preference on v, user u is influenced by personal interest with probability λ_u, and is influenced by the local preference of l_u with probability $1 - \lambda_u$. It is worth mentioning that LCA–LDA holds personalized mixing weights for individual users, considering the differences between users in personality (e.g. openness, agreeableness).

To further alleviate the data sparsity problem, LCA–LDA incorporates the content information of spatial items. Thus, we reformulate Eq. (3.1) as follows:

$$P(v|\theta_u, \theta'_{l_u}, \phi, \phi') = \sum_{c \in C_v} P(v, c|\theta_u, \theta'_{l_u}, \phi, \phi') \tag{3.2}$$

$$P(v|\theta_u, \phi, \phi') = \sum_{c \in C_v} P(v, c|\theta_u, \phi, \phi') \tag{3.3}$$

$$P(v|\theta'_{l_u}, \phi, \phi') = \sum_{c \in C_v} P(v, c|\theta'_{l_u}, \phi, \phi') \tag{3.4}$$

where C_v is a set of content words describing spatial item v. In LCA–LDA, both user interest θ_u and local preference θ'_{l_u} are modeled by a multinomial distribution over latent topics. Each spatial item v is generated from a sample topic z. LCA–LDA also parameterizes a distribution over content words associated with each topic z, and thus topics are responsible for simultaneously generating both spatial items and their content words. As shown in Fig. 3.2, the weight (a, b) on the edge linking topic $T1$ and item $v2$ represents the probabilities of $T1$ generating item $v2$ and its associated content word "Sculptures," respectively. It should be noted that here we assume that items and their content words are independently conditioned on the topics. So, $P(v, c|\theta_u, \phi, \phi')$ and $P(v, c|\theta'_{l_u}, \phi, \phi')$ can be computed according to Eqs. (3.5) and (3.6). Parameter estimation in LCA–LDA is thus driven to discover topics that capture both item cooccurrence and content word cooccurrence patterns. This encodes our prior knowledge that spatial items having many common users or similar content should be clustered into the same topic with high probability.

$$\begin{aligned} P(v, c|\theta_u, \phi, \phi') &= \sum_z P(v, c|z, \phi_z, \phi'_z) P(z|\theta_u) \\ &= \sum_z P(v|z, \phi_z) P(c|z, \phi'_z) P(z|\theta_u) \end{aligned} \tag{3.5}$$

$$\begin{aligned} P(v, c|\theta'_{l_u}, \phi, \phi') &= \sum_z P(v, c|z, \phi_z, \phi'_z) P(z|\theta'_{l_u}) \\ &= \sum_z P(v|z, \phi_z) P(c|z, \phi'_z) P(z|\theta'_{l_u}) \end{aligned} \tag{3.6}$$

The proposed LCA–LDA is a latent class statistical mixture model. It can be represented by a graphical model in Fig. 3.3 and a generative process in Algorithm 1. The model discovers (1) user's personal interest distribution over latent topics, θ_u; (2) local preference distribution over latent topics, θ'_{l_u}; (3) topic distribution over items, ϕ_z; (4) topic distribution over content words, ϕ'_z; (5) the mixing weight λ_u. The generative model aims to capture the process of human behaviors and/or reasoning for decision making. For example, a querying user u wants to choose a venue v in

Algorithm 1: Probabilistic generative process in LCA–LDA

Input: a user profile dataset D;
Output: estimated parameters θ, θ', ϕ, ϕ', and λ;

for *each topic* z **do**
  Draw $\phi_z \sim Dirichlet(\cdot|\beta)$;
  Draw $\phi'_z \sim Dirichlet(\cdot|\beta')$;
end
for *each* D_u *in* D **do**
  for *each record* $(u, v_{ui}, l_{ui}, c_{ui}) \in D_u$ **do**
    Toss a coin s_{ui} according to $bernoulli(s_{ui}) \sim beta(\gamma, \gamma')$;
    if $s_{ui} = 1$ **then**
      Draw $\theta_u \sim Dirichlet(\cdot|\alpha)$;
      Draw a topic $z_{ui} \sim multi(\theta_u)$ according to the interest of user u;
    end
    if $s_{ui} = 0$ **then**
      Draw $\theta'_{l_{ui}} \sim Dirichlet(\cdot|\alpha')$;
      Draw a topic $z_{ui} \sim multi(\theta'_{l_{ui}})$ according to the local preference of l_{ui};
    end
    Draw an item $v_{ui} \sim multi(\phi_{z_{ui}})$ from z_{ui}-specific spatial item distribution;
    Draw a content word $c_{ui} \sim multi(\phi'_{z_{ui}})$ from z_{ui}-specific content word distribution;
  end
end

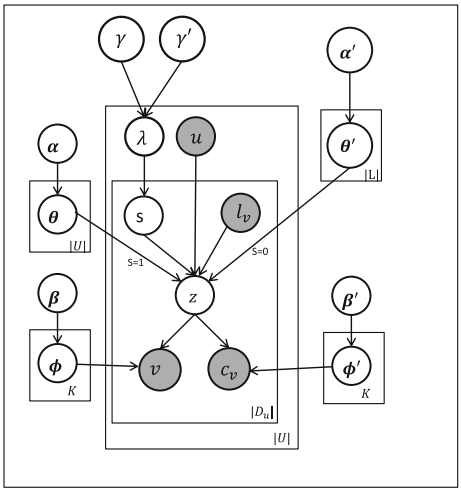

Fig. 3.3 The graphical representation of LCA–LDA [21] © 2014 Association for Computing Machinery, Inc. Reprinted by permission

city l_u to visit. The person may choose one based on personal interest or choose the one that is most popular in l_u. In the case that u wants to choose the venue based on personal interest (with a certain probability λ_u), a topic z is first chosen according to the personal interest distribution θ_u, and then the selected topic z in turn generates a venue v and relevant content words C_v following on the topic's item and content word generative distributions (i.e., ϕ_z and ϕ'_z), respectively. In the

case that user u follows the local preference of l_u, l_u would generate an item and its content words following on l_u's preference distribution θ'_{l_u} similarly. Thus, this model simulates the process that how u picks the spatial item v, including how the local preference of l_u influences u's decision. As a running example in Fig. 3.2, the user is influenced by personal interest and the local preference with probabilities 0.64 and 0.36, respectively. The top-4 topics of the user's interest and the local preference are also shown, respectively, where the weights representing user's personal interest and the local preference in the topics are labeled in the corresponding edges. We can see that there is only one overlapped topic for the user and the local preference, and their dominated topics are different (i.e., T1:0.801 vs. T8:0.756). The probabilities of topic generating items and their associated content words are also labeled in the corresponding edges. For example, the weights (a, b) on the edge linking topic T1 and item v2 represent the probabilities of T1 generating item v2 and its associated content word "Sculptures", respectively.

With the model hyperparameters α, α', β, β', γ, and γ', the joint distribution of the observed and hidden variables $\boldsymbol{v}$, $\boldsymbol{c}_v$, $\boldsymbol{z}$ and $\boldsymbol{s}$ can be written as below.

$$\begin{aligned} & P(\boldsymbol{v}, \boldsymbol{c}_v, \boldsymbol{z}, \boldsymbol{s} | \alpha, \alpha', \beta, \beta', \gamma, \gamma') \\ & = \int \cdots \int P(\boldsymbol{v}|\phi, \boldsymbol{z}) P(\phi|\beta) P(\boldsymbol{c}_v|\phi', \boldsymbol{z}) P(\phi'|\beta') \\ & \quad P(\boldsymbol{z}|\theta, \theta', \boldsymbol{s}) P(\theta|\alpha) P(\theta'|\alpha') P(\boldsymbol{s}|\lambda) P(\lambda|\gamma, \gamma') \\ & \quad d\phi \, d\phi' \, d\theta \, d\theta' \, d\lambda \end{aligned} \tag{3.7}$$

3.2.3 Model Inference

The computation of the posterior distribution of the hidden variables is intractable for the LCA–LDA model. Therefore, we follow the studies [17, 18] and use approximate method collapsed Gibbs sampling to obtain samples of the hidden variable assignment and to estimate unknown parameters $\{\theta, \theta', \phi, \phi', \lambda\}$ in the LCA–LDA. As for the hyperparameters α, α', β, β', γ, and γ', for simplicity, we take a fixed value (i.e., $\alpha = \alpha' = 50/K$, $\beta = \beta' = 0.01$, $\gamma = \gamma' = 0.5$). Note that Gibbs sampling allows the learning of a model by iteratively updating each latent variable given the remaining variables. In the sampling procedure, we begin with the joint probability of all user profiles in the dataset. Next, using the chain rule, we obtain the posterior probability of sampling topics for each four-tuple (u, v, l_v, c_v). Specifically, we employ a two-step Gibbs sampling procedure.

To begin with, we need to compute the conditional probabilities $P(s_{ui}|\boldsymbol{s}_{\neg ui}, \boldsymbol{z}, \boldsymbol{v}, \boldsymbol{c})$ and $P(z_{ui}|\boldsymbol{s}, \boldsymbol{z}_{\neg ui}, \boldsymbol{v}, \boldsymbol{c})$, where $\boldsymbol{s}_{\neg ui}$ and $\boldsymbol{z}_{\neg ui}$ represent the s and z assignments for all the spatial items except v_{ui}, respectively. 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 Eq. (3.7). Next, to make the sampling procedure clearer, we factorize this joint probability as:

$$P(\boldsymbol{v}, \boldsymbol{c_v}, z, s) = P(\boldsymbol{v}|z)P(\boldsymbol{c_v}|z)P(z|s)P(s) \tag{3.8}$$

By integrating out the parameter ϕ in Eq. (3.7) we can obtain the first term in Eq. (3.8):

$$P(\boldsymbol{v}|z) = \left(\frac{\Gamma(\sum_v \beta_v)}{\prod_v \Gamma(\beta_v)}\right)^K \prod_z \frac{\prod_v \Gamma(n_{zv} + \beta_v)}{\Gamma(\sum_v (n_{zv} + \beta_v))} \tag{3.9}$$

where n_{zv} is the number of times that spatial item v has been generated by topic z. $\Gamma(\cdot)$ is the gamma function. Similarly, for the second term $P(\boldsymbol{c_v}|z)$ in Eq. (3.8), we integrate out the parameter ϕ' and get

$$P(\boldsymbol{c_v}|z) = \left(\frac{\Gamma(\sum_c \beta'_c)}{\prod_c \Gamma(\beta'_c)}\right)^K \prod_z \frac{\prod_c \Gamma(n_{zc} + \beta'_c)}{\Gamma(\sum_c (n_{zc} + \beta'_c))} \tag{3.10}$$

where n_{zc} is the number of times that content word c has been generated by topic z.

Next, we evaluate the third term $P(z|s)$ in Eq. (3.8). By integrating out the parameters θ_u and θ'_l, we compute:

$$\begin{aligned} P(z|s) = &\left(\frac{\Gamma(\sum_z \alpha_z)}{\prod_z \Gamma(\alpha_z)}\right)^{|U|} \prod_u \frac{\prod_z \Gamma(n_{uz} + \alpha_z)}{\Gamma(\sum_z (n_{uz} + \alpha_z))} \\ &\cdot \left(\frac{\Gamma(\sum_z \alpha'_z)}{\prod_z \Gamma(\alpha'_z)}\right)^{|L|} \prod_l \frac{\prod_z \Gamma(n_{lz} + \alpha'_z)}{\Gamma(\sum_z (n_{lz} + \alpha'_z))} \end{aligned} \tag{3.11}$$

where $|U|$ is the number of users, and $|L|$ is the number of locations (e.g., cities); n_{uz} is the number of times that topic z has been sampled from the multinomial distribution specific to user u; n_{lz} is the number of times that topic z has been sampled from the multinomial distribution specific to location l.

Last, we need to derive the fourth term $P(s)$. By integrating out λ_u we have:

$$P(s) = \left(\frac{\Gamma(\gamma + \gamma')}{\Gamma(\gamma)\Gamma(\gamma')}\right)^{|U|} \prod_u \frac{\Gamma(n_{us_1} + \gamma)\Gamma(n_{us_0} + \gamma')}{\Gamma(n_{us_1} + n_{us_0} + \gamma + \gamma')} \tag{3.12}$$

where n_{us_1} is the number of times that $s = 1$ has been sampled in the user profile D_u; n_{us_0} is the number of times that $s = 0$ has been sampled in the user profile D_u.

Now, the conditional probability can be obtained by multiplying and canceling of terms in Eqs. (3.9)–(3.12). Thus, we first sample the coin s according to the posterior probability:

$$\begin{aligned} &P(s_{ui} = 1|\boldsymbol{s}_{\neg ui}, \boldsymbol{z}, .) \\ &\propto \frac{n^{\neg ui}_{uz_{ui}} + \alpha_{z_{ui}}}{\sum_z (n^{\neg ui}_{uz} + \alpha_z)} \times \frac{n^{\neg ui}_{us_1} + \gamma}{n^{\neg ui}_{us_0} + n^{\neg ui}_{us_1} + \gamma + \gamma'} \end{aligned} \tag{3.13}$$

$$P(s_{ui} = 0|\boldsymbol{s}_{\neg ui}, \boldsymbol{z}, .) \propto \frac{n^{\neg ui}_{l_{ui}z_{ui}} + \alpha'_{z_{ui}}}{\sum_z (n^{\neg ui}_{l_{ui}z} + \alpha'_z)} \times \frac{n^{\neg ui}_{us_0} + \gamma'}{n^{\neg ui}_{us_0} + n^{\neg ui}_{us_1} + \gamma + \gamma'} \tag{3.14}$$

where the number $n^{\neg ui}$ with superscript $\neg ui$ denotes a quantity, excluding the current instance.

Then, we sample topic z according to the following posterior probability, when $s_{ui} = 1$:

$$P(z_{ui}|s_{ui} = 1, \boldsymbol{z}_{\neg ui}, \boldsymbol{v}, \boldsymbol{c}, .) \propto \frac{n^{\neg ui}_{uz_{ui}} + \alpha_{z_{ui}}}{\sum_z (n^{\neg ui}_{uz} + \alpha_z)} \frac{n^{\neg ui}_{z_{ui}v_{ui}} + \beta_{v_{ui}}}{\sum_v (n^{\neg ui}_{z_{ui}v} + \beta_v)} \frac{n^{\neg ui}_{z_{ui}c_{ui}} + \beta'_{c_{ui}}}{\sum_c (n^{\neg ui}_{z_{ui}c} + \beta'_c)} \tag{3.15}$$

when $s_{ui} = 0$:

$$P(z_{ui}|s_{ui} = 0, \boldsymbol{z}_{\neg ui}, \boldsymbol{v}, \boldsymbol{c}, .) \propto \frac{n^{\neg ui}_{l_{ui}z_{ui}} + \alpha'_{z_{ui}}}{\sum_z (n^{\neg ui}_{l_{ui}z} + \alpha'_z)} \frac{n^{\neg ui}_{z_{ui}v_{ui}} + \beta_{v_{ui}}}{\sum_v (n^{\neg ui}_{z_{ui}v} + \beta_v)} \frac{n^{\neg ui}_{z_{ui}c_{ui}} + \beta'_{c_{ui}}}{\sum_c (n^{\neg ui}_{z_{ui}c} + \beta'_c)} \tag{3.16}$$

After a sufficient number of sampling iterations, the approximated posterior can be used to get estimates of parameters by examining the counts of (s, z) assignments to four-tuple (u, v, l, c). Specifically, during the parameter estimation, the algorithm keeps track of a $K \times |V|$ (topic by spatial item) count matrix, a $K \times |C|$ (topic by content word) count matrix, an $|U| \times 2$ (user by coin) count matrix, an $|U| \times K$ (user by topic) count matrix and an $|L| \times K$ (location by topic) count matrix. Given these matrices, we can estimate the parameters θ, θ', ϕ, ϕ', and λ as follows:

$$\hat{\theta}_{uz} = \frac{n_{uz} + \alpha_z}{\sum_{z'} (n_{uz'} + \alpha_{z'})} \tag{3.17}$$

$$\hat{\theta}'_{lz} = \frac{n_{lz} + \alpha'_z}{\sum_{z'} (n_{lz'} + \alpha'_{z'})} \tag{3.18}$$

$$\hat{\phi}_{zv} = \frac{n_{zv} + \beta_v}{\sum_{v'} (n_{zv'} + \beta_{v'})} \tag{3.19}$$

$$\hat{\phi}'_{zc} = \frac{n_{zc} + \beta'_c}{\sum_{c'} (n_{zc'} + \beta'_{c'})} \tag{3.20}$$

$$\hat{\lambda}_u = \frac{n_{us_1} + \gamma}{n_{us_1} + n_{us_0} + \gamma + \gamma'} \tag{3.21}$$

3.2.4 Online Recommendation

In this subsection, we present the online recommendation part of our recommender system LCARS. Given a querying user u with a querying region l_u to which u is going to travel, the online part of LCARS will efficiently compute ranking scores for all spatial items within l_u and then return the top-k items as the recommendation results.

The ranking scores of spatial items are computed using the knowledge, such as user interest θ, local preference θ', mixing weight λ, topics ϕ, and ϕ', learned by the offline model LCA–LDA. To improve the online query performance, we propose a ranking framework in Eq. (3.22) which separates the offline scoring computation from the online scoring computation. Specifically, $F(l_u, v, z)$ represents the offline part of the scoring, denoting the score of spatial item v with respect to location l_u on topic z which is learnt in the LCA-LDA model. Note that $F(l_u, v, z)$ is independent of querying users. The weight score $W(u, l_u, z)$ is computed in the online part, denoting the preference weight of query (u, l_u) on topic z. It is worth mentioning that the main time-consuming components of $W(u, l_u, z)$ are also computed offline (e.g., $\hat{\theta}_{uz}$, λ_u, and $\hat{\theta}'_{l_u z}$), and the online computation is just a simple linear fusion process, as is shown in Eq. (3.23). This design enables maximum precomputation for the problem considered, and in turn minimizes the query time. At query time, the ranking score $S(u, l_u, v)$ in Eq. (3.22) only needs to aggregate $F(l_u, v, z)$ over K topics by a simple weighted sum function, in which the weight is $W(u, l_u, z)$. From Eqs. (3.23) and (3.24), we can see that $W(u, l_u, z)$ consists of two components, designed to model user interest and local preference, respectively, and each component is associated with a kind of user motivation. $F(l_u, v, z)$ takes into account both the item co-occurrence information and the similarity of item contents to produce recommendations.

$$S(u, l_u, v) = \sum_z F(l_u, v, z) W(u, l_u, z) \tag{3.22}$$

$$W(u, l_u, z) = \hat{\lambda}_u \hat{\theta}_{uz} + (1 - \hat{\lambda}_u) \hat{\theta}'_{l_u z} \tag{3.23}$$

$$F(l_u, v, z) = \begin{cases} \hat{\phi}_{zv} \sum_{c_v \in C_v} \hat{\phi}'_{z c_v} & v \in V_{l_u} \\ 0 & v \notin V_{l_u} \end{cases} \tag{3.24}$$

3.3 Experiments

In this section, we first describe the settings of experiments including the datasets, comparative approaches, and the evaluation method. We then report major experimental results on both the recommendation effectiveness and efficiency of our recommender system, followed by their tradeoff. We also study the interpretability of

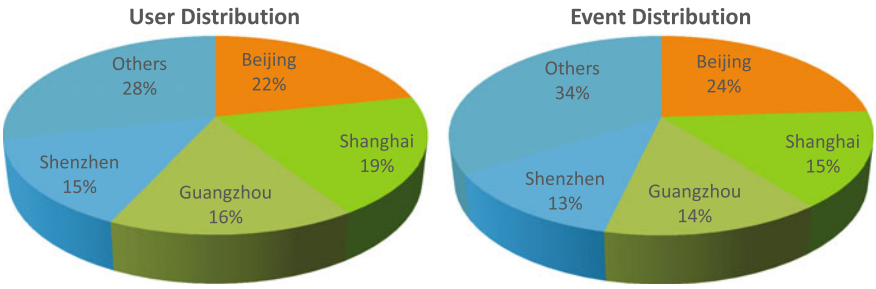

Fig. 3.4 User and Event distributions over cities

our LCARS by analyzing the learnt user profiles, the effect of users' personal interests and the local preferences in users' decision making for traveling, and the latent topics learnt by LCARS.

3.3.1 Datasets

In this chapter, we utilize one synthetic dataset with 10 million spatial items and two real-life datasets for the performance evaluation. The detailed description for two real datasets are listed as follows.

- **DoubanEvent**: DoubanEvent is China's largest event-based social networking site where users can publish and participate in social events. On DoubanEvent, a social event is created by a user by specifying what, when and where the event is. Other users can express their intent to join events by checking-in online. This dataset consists of 100,000 users, 300,000 events and 3,500,000 check-ins. Most of check-in records are located in China's four largest cities: Beijing, Shanghai, Guangzhou and Shenzhen. To guarantee the validity of the experimental results, each user in our dataset has provided at least 10 check-ins. Figure 3.4 describes the distribution information of both users and events over cities. For instance, 22 % of users live in the city of Beijing and 24 % of events are held in Beijing. The following information is recorded when collecting the data: (1) user information, including user-id, user-name, and user-home city; (2) event information, consisting of event-id, event-name, event-latitude, event-longitude, event-summary, and its category; (3) user feedback information, including user-id and event-id. We make the dataset publicly available.[1]
- **Foursquare**: Another publicly available LBSNs dataset, Foursquare [6], is also used in our experiment. Foursquare is one of the most popular online LBSNs. It has more than 30 million users and 3 billion check-ins as of January, 2013.[2]

[1] https://sites.google.com/site/dbhongzhi/.

[2] https://foursquare.com/about/.

The web site itself does not provide a public API to access users' check-in data; however, it provides an alternative way for users to link their twitter accounts with Foursquare, and then share the check-in message as tweets to Twitter. This dataset contains 11,326 users, 182,968 venues, and 1,385,223 check-ins. Note that this dataset does not contain item content information.

To utilize these two datasets in our proposed models, we preprocess them as follows: (1) We first employ Google Maps API[3] to partition all the spatial items into cities according to their latitudes and longitudes. (2) For the DoubanEvent dataset, we then use NLP toolkits[4] to extract a set of content words for each event from its summary and category description. To guarantee the quality of content words, we use tf-idf techniques to rank all content words associated with each event and finally keep top five ranked ones.

Note that, although we only utilize the city granularity to generate recommendation to end-users for evaluation in this chapter, our approach can be easily extended to facilitate the recommendation task at various granularities, by dividing the space into multiscale regions, inferring their local preferences offline, and automatically selecting proper region when making recommendations.

3.3.2 Comparative Approaches

We compare our proposed LCARS with the following six competitor methods, where the first four approaches are the existing recommender systems, and the last two recommender models correspond to the two main components of our proposed LCA–LDA.

- **User interest, social, and geographical influences (USG)**: Following recent location-based recommendation work [19], a unified location recommendation framework is implemented which linearly fuses user interest, along with the social and geographical influences. The user interest component of USG is implemented by a traditional user-based collaborative filtering technique, and the geographical influence is computed by a power-law probabilistic model that aims to capture the *geographical clustering phenomenon* that points of interest visited by the same user tend to be clustered geographically.
- **Social Trust Ensemble (STE)**: Social Trust Ensemble, proposed in [13], is a probabilistic matrix factorization framework which linearly fuses the users' tastes and their friends' favors together to produce recommendations. It should be noted that the mixing weights in STE are manually set rather than learnt automatically from the data. Besides, the mixing weights in BTE are not personalized (i.e., all users in a dataset share the same mixing weights), ignoring the differences between users.

[3] https://developers.google.com/maps/.

[4] http://nlp.stanford.edu/software/index.shtml.

- **Category-based k-Nearest Neighbors Algorithm (CKNN)**: Following recently proposed location-based recommendation technique [2] for dealing with the problem of data sparsity, a category-based KNN algorithm is implemented as our competitor. CKNN first projects a user's activity history into a well-designed category space and models each user's preference with a weighted category hierarchy. Meanwhile, it infers the authority of each user in a city with respect to different category of spatial items according to their activity histories using HITS model. When receiving a query $q = (u, l)$, CKNN first selects a set of high-quality users N_u in the querying city who have the same or similar preferences with the querying user u. Then CKNN constructs a user-item matrix using the selected users N_u and their visited spatial items. Finally, CKNN employs a traditional user-based CF model over the user-item matrix to infer the querying user's rating of a candidate item. The general intuition behind a user-based CF model is that similar users rate the same items similarly. Formally, the rating that the querying user u would give to spatial item v is calculated as follows.

$$S(u, v) = \sum_{u' \in N_u} Sim(u, u') \times r(u', v) \tag{3.25}$$

  where $Sim(u, u')$ denotes the similarity between u and u' which is computed according to their weights in the category hierarchy rather than the traditional Cosine value between two users' item vectors; $r(u', v)$ represents the rating that u' gave to item v.
- **Item-based k-Nearest Neighbors Algorithm (IKNN)**: IKNN is the most common way that people come up with, which applies the collaborative filtering method directly over the spatial items. This method utilizes the user activity history to create a user-item matrix. When receiving a query, IKNN retrieves all users to find k nearest neighbors in the querying city by computing the Cosine similarity between the querying user's and other users' item vectors. Finally, the spatial items in the user-specific querying city that have a relatively high ranking score will be recommended. It should be noted that when IKNN cannot help the querying user find k nearest neighbors in the querying city, we recommend the most popular local ones.
- **LDA**: Following previous works [3, 8], a standard LDA-based method is implemented as one of our baselines. In this model, each user is viewed as a document, and spatial items visited by the user are viewed as the words appeared in the document. Compared with our proposed LCA–LDA, this method neither considers the content information of spatial items, nor their location information. For online recommendation, the ranking score is computed using our ranking framework in Eq. (3.22) where $F(l_u, v, z) = \hat{\phi}_{zv}$, $W(u, l_u, z) = \hat{\theta}_{uz}$.
- **Location-Aware LDA (LA–LDA)**: As a component of the proposed LCA–LDA model, LA–LDA means our method without considering the content information of spatial items. For online recommendation, the ranking score is computed using our proposed ranking framework in Eq. (3.22) where $F(l_u, v, z) = \hat{\phi}_{zv}$ and $W(u, l_u, z) = \hat{\lambda}_u \hat{\theta}_{uz} + (1 - \hat{\lambda}_u)\hat{\theta}'_{l_u z}$.

- **Content-Aware LDA (CA–LDA)**: As another component of the LCA–LDA model, CA–LDA means our method without exploiting the location information of spatial items, i.e., local preference. It can capture the prior knowledge that spatial items with the same or similar contents are more likely to belong to the same topic. This model is similar to the ACT model [18] in the methodology. For online recommendation, the ranking score is computed using our ranking framework in Eq. (3.22) where $F(l_u, v, z) = \hat{\phi}_{zv} \sum_{c_v \in C_v} \hat{\phi}'_{zc_v}$ and $W(u, l_u, z) = \hat{\theta}_{uz}$.

3.3.3 Evaluation Methods

To make an overall evaluation of the recommendation effectiveness of our proposed LCA–LDA, we first design the following two real settings: (1) querying cities are new cities to querying users; (2) querying cities are the home cities of querying users. We then divide a user's activity history into a test set and a training set. We adopt two different dividing strategies with respect to the two settings. For the first setting, we select all spatial items visited by the user in a non-home city as the test set and use the rest of the user's activity history in other cities as the training set. For the second setting, we randomly select 20 % of spatial items visited by the user in personal home city as the test set, and use the rest of personal activity history as the training set.

According to the above designed dividing strategies, we split the user activity history S into the training dataset $S_{training}$ and the test set S_{test}. To evaluate the recommender models, we adopt the similar testing methodology and the measurement Recall@k applied in [3, 5, 9, 20]. Specifically, for each test case (u, v, l_v) in S_{test}:

1. We compute the ranking score for item v as well as all other spatial items located in city l_v and unvisited by u before.
2. We form a ranked list by ordering all these spatial items according to their ranking scores. Let p denote the rank of the test item v within this list. The best result corresponds to the case where the test item v precedes all the unvisited items (i.e., $p = 0$).
3. We form a top-k recommendation list by picking the k top ranked items from the list. If $p < k$ we have a hit (i.e., the test item v is recommended to the user). Otherwise we have a miss. The probability of a hit increases with the increasing value of k. When k is equal to the number of spatial items located in l_v, we always have a hit.

The computation of Recall@k proceeds as follows. We define hit@k for a single test case as either the value 1 if the test spatial item v appears in the top-k results, or else the value 0. The overall Recall@k are defined by averaging all test cases:

$$Recall@k = \frac{\#hit@k}{|S_{test}|} \tag{3.26}$$

where $\#hit@k$ denotes the number of hits in the test set, and $|S_{test}|$ is the number of all test cases. It should be noted that both DoubanEvent and Foursquare datasets have a low density, which usually results in relatively low recall values. In addition, the spatial items visited by user u may represent only a small portion of spatial items that u is interested in, so the hypothesis that all the unvisited spatial items are nonrelevant to user u tends to underestimate the computed recall with respect to the true recall. However, this experimental setting and evaluation are fair to all comparison approaches. So, we focus on the relative improvements we achieve, instead of the absolute values in this chapter.

3.3.4 Recommendation Effectiveness

In this part, we first present the optimal performance with well-tuned parameters and then study the impact of model parameters.

Figure 3.5 reports the performance of the recommendation algorithms on DoubanEvent dataset. We show only the performance where k is in the range $[1 \ldots 20]$, because a greater value of k is usually ignored for a typical top-k recommendation task. It is apparent that the algorithms have significant performance disparity in terms of top-k recall. As shown in Fig. 3.5a where querying cities are new cities, the recall of LCA–LDA is about 0.1 when $k = 10$, and 0.126 when $k = 20$ (i.e., the model has a probability of 10 % of placing an appealing event within the querying city in the top-10 and 12.6 % of placing it in the top-20). Clearly, our proposed LCA–LDA model outperforms other competitor recommendation algorithms significantly. First, IKNN and USG perform the worst in the new city setting. Both of them are traditional location-based recommendation algorithms and cannot alleviate the data sparsity problem in new cities. Specifically, without exploiting the content/category information of spatial items to build a bridge, they cannot transfer the users' prefer-

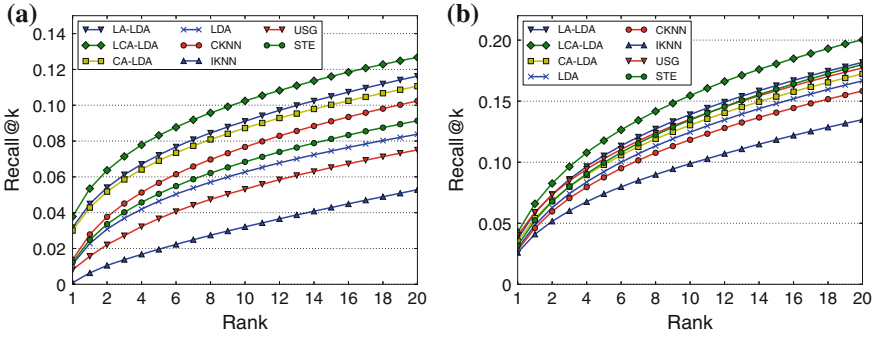

Fig. 3.5 Top-k performance on DoubanEvent. **a** Users traveling in new cities. **b** Users traveling in home cities

ences learnt in home cities to new cities, and hence fail to find k preference-similar users for the querying user in the new city setting. Second, STE and USG perform better than LDA and IKNN, respectively, due to the benefits brought by considering social influence from friends. Besides, STE and LDA outperform both USG and IKNN consistently, showing the advantages of latent factor models which overcome the data sparsity problem, to some extent, by dimension reduction. But STE performs worse than CKNN and our LCA–LDA, which shows that exploiting social influences and dimension reduction are not enough to alleviate the *new city* problem although they can alleviate the data sparsity problem to some extent. As is analyzed in [4], most of a querying user's friends live in the same city with the querying user, and they also have few footprints in the querying city that is new to the querying user due to the property of travel locality. That is why exploiting social and geographical influence cannot help much when alleviating the *new city* problem. Third, CKNN, which was proposed for solving the *new city* problem [2], performs better than STE, IKNN, USG, and LDA, as is expected. CKNN depends on a well-designed category hierarchy to facilitate users' preferences across cities. So, it can find k high-quality users who have similar/same preferences with the querying user. But, this method ignores the observation of the shift of users' preferences, i.e., people's preferences or behavioral patterns may change when they travel in different cities, especially in cities that are new to them. So, CKNN would fail to make accurate recommendations in the case where users' preferences shift, while our proposed LCA–LDA and LA-LDA models can still work well in this case because they exploit the local preferences/attractions of the querying city to produce recommendations, i.e., what most of people have done when they travel in the querying city. That is why our proposed LCA-LDA model performs much better than CKNN. Fourth, LA-LDA outperforms LDA, justifying the benefit brought by considering local preferences, and CA-LDA exceeds LDA due to the advantages of taking item contents into consideration. Finally, LCA-LDA outperforms both LA-LDA and CA-LDA, showing the advantages of combining local preferences and item contents in a unified manner.

In Fig. 3.5b, we report the performance of all recommendation algorithms for the second setting where querying cities are home cities of querying users. From the figure, we can see that the trend of comparison result is similar to that presented in Fig. 3.5a. The main differences are that (1) all recommendation algorithms perform better in the home city setting than in the new city setting and (2) the performance gaps between different recommendation methods narrow, because the data sparsity problem is not so severe in the home city setting. Another observation is that USG and STE almost performs as well as LA–LDA, and outperforms LDA, CKNN, and IKNN in the home city setting, verifying the benefit brought by considering the social and geographical influences. However, the performances of USG and STE are not so well in the new city setting, as shown in Fig. 3.5a, which shows that exploiting social and geographical influences is not enough to alleviate the *new city* problem although it can alleviate the data sparsity problem to some extent. The third observation is that CKNN outperforms LDA and STE in Fig. 3.5a while LDA and STE slightly

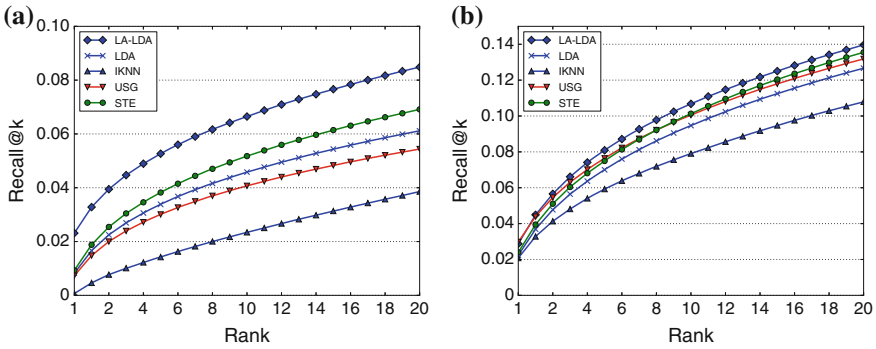

Fig. 3.6 Top-k performance on Foursquare. **a** Users traveling in new cities. **b** Users traveling in home cities

exceeds CKNN in Fig. 3.5b, showing that existing latent factor models (e.g., LDA and STE) better suit the home city setting which is almost the same as the traditional recommendation setting (e.g., movie recommendation), and the hybrid methods (e.g., CKNN) are more capable of overcoming the difficulty of data sparsity, i.e., the *new city* problem.

Figure 3.6 reports the performance of the recommendation algorithms on the Foursquare dataset. We only compare LA–LDA, one component of our LCA–LDA model, with LDA, STE, USG, and IKNN since this dataset does not contain item content information. From the figure, we can see that the trend of comparison result is similar to that presented in Fig. 3.5, and LA–LDA performs best, showing the advantage of exploiting the local preference.

3.3.5 Local Preference Influence Study

In this section, we study the effects of personal interest and local preference on users' decision making. The self interest influence probability λ_u and the local preference influence probability $1 - \lambda_u$ are learnt automatically in our proposed LCA–LDA model. Since different people have different mixing weights, we plot the distributions of both self interest and local preference influence probabilities among all users. The results on the DoubanEvent dataset are shown in Fig. 3.7, where Fig. 3.7a plots the cumulative distribution of self interest influence probabilities, and Fig. 3.7b shows the local preference influence probabilities. It can be observed that, in general, people's self interest influence is higher than the influence of the local preference. For example, Fig. 3.7a shows that the self interest influence probability of more than 70 % of users is higher than 0.5. The implication of this finding is that people mainly attend social

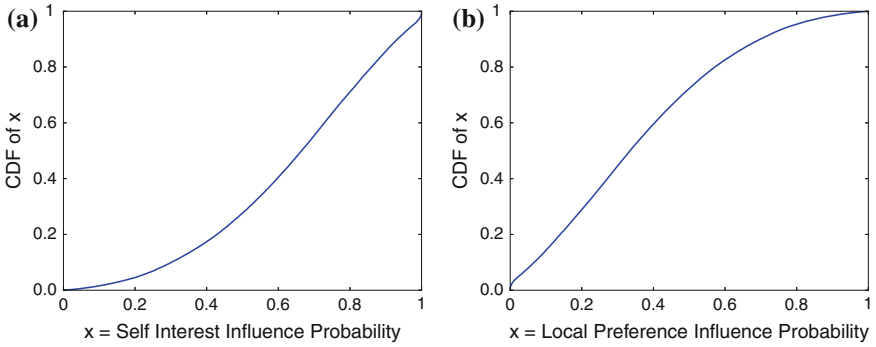

Fig. 3.7 Local preference influence result (DoubanEvent). **a** Self interest influence. **b** Local preference influence

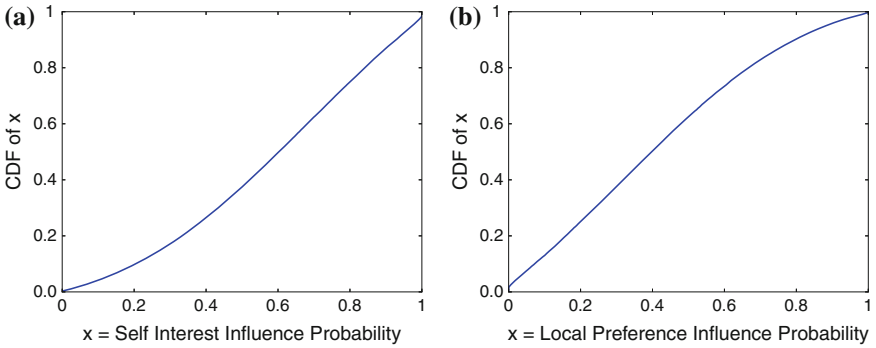

Fig. 3.8 Local preference influence result (Foursquare). **a** Self interest influence. **b** Local preference influence

events based on their self interests, but they sometimes attend popular local events regardless of their interests, especially when traveling in new cities. This finding also explains the superiority of LCA–LDA and LA–LDA in the recommendation performance (Sect. 4.2.1).

Figure 3.8a, b show, respectively, the self interest influence probabilities and local preference influence probabilities learnt from the Foursquare data by LA–LDA model. We observe that the trend of the CDF curve in Fig. 3.8 is similar to that in Fig. 3.7. As shown in Fig. 3.8, although the self-interest influence probability is generally higher than that of the local preference, the local preference still plays an important role in the user decision-making for visiting. For example, the local preference influence probability of about 40 % of users is higher than 0.5. This finding also shows the necessity of exploiting the local preference in spatial item recommendation.

Table 3.2 Latent topics learned by LCA–LDA

T8			T22			T9		
Event ID	Category	Location	Event ID	Category	Location	Event ID	Category	Location
18852778	Seminar	Beijing	14232509	Exhibition	Beijing	18567203	Concert	Shanghai
11177738	Seminar	Shanghai	18833193	Exhibition	Shanghai	18131435	Concert	Beijing
18845712	Seminar	Nanjing	18761132	Exhibition	Beijing	18898584	Concert	Shanghai
18833831	Seminar	Beijing	18619185	Exhibition	Xi'an	18825734	Concert	Shanghai
18129058	Seminar	Wuhan	18818656	Exhibition	Shanghai	18710070	Concert	Guangzhou
18840452	Seminar	Beijing	18696716	Exhibition	Beijing	18465268	Concert	Chengdu
18867591	Seminar	Guangzhou	18800412	Exhibition	Beijing	18631346	Concert	Beijing
18953054	Seminar	Shanghai	12104434	Exhibition	Shanghai	18394935	Concert	Shanghai

Table 3.3 Latent topics learned by LDA

T9			T8			T16		
Event ID	Category	Location	Event ID	Category	Location	Event ID	Category	Location
18825734	Concert	Shanghai	18852778	Seminar	Beijing	18020482	Exhibition	Guangzhou
18830050	Film	Shanghai	18840452	Seminar	Beijing	18425473	Concert	Guangzhou
18818656	Exhibition	Shanghai	18629384	Film	Beijing	18847061	Seminar	Guangzhou
16578267	Film	Shanghai	18432390	Drama	Beijing	18937837	Party	Guangzhou
18567203	Concert	Shanghai	18668341	Concert	Beijing	18847604	Film	Guangzhou
17364244	Drama	Shanghai	18041992	Exhibition	Beijing	18829026	Concert	Guangzhou
18053337	Concert	Shanghai	18953054	Seminar	Beijing	18412853	Drama	Guangzhou
13892914	Culture salon	Shanghai	18478314	Drama	Beijing	17364134	Concert	Guangzhou

3.3.6 Analysis of Latent Topic

To analyze why our proposed location-content-aware probabilistic generative model LCA–LDA performs better than LDA in the task of top-k recommendation, especially spatial item recommendation in new cities, we investigate the latent information learnt from LCA–LDA and LDA.

Tables 3.2 and 3.3, respectively, show three latent topics (e.g., T8, T22, and T9) learned by LCA–LDA and LDA on the DoubanEvent dataset. For each topic, we present the top eight events with the highest probabilities, including their IDs, categories and locations. For locations, we present only the cities rather than detailed addresses because of space constraints. Each event can be browsed online by combining event-ID and the prefix URL.[5] For example, the event 18852778 can be accessed at www.douban.com/event/11177738/. By comparing the topics in Tables 3.2 and 3.3, we observe that the events in each latent topic learned by LCA–LDA not only share the same category, but are also located in different cities. In contrast, the topics learnt by LDA are not category-consistent. For example, concerts, culture salons,

[5] http://www.douban.com/event/.

films and exhibitions are mixed up in the topics T9, T8, and T16 learnt by LDA. Besides, the events in each topic learnt by LDA take place in the same city. For example, the events in topic T9 are located in Shanghai and the events in topic T8 are held in Beijing.

The comparative results reveal that when we use existing topic model LDA to analyze the user activity history, we are unable to discover the users' interests in the features (latent topics) of spatial items such as "exhibition" and "concert", and most of the estimated topics describe the user's spatial area of activity instead of user interests. That is because (1) the user's choice of spatial items is largely influenced by her/his geographical coordinates, and spatial items in the user's immediate neighborhood are likely to be chosen; (2) traditional latent factor models (e.g., topic models and matrix factorization methods) aim to capture item cooccurrence patterns. Another finding is that exploiting the content information of spatial items facilitates the clustering of items which are not only category-alike but also geodiversity, alleviating the *new city* problem. The experimental results also explains the superiority of LCA–LDA and CA–LDA in the recommendation performance (Sect. 3.3.4).

3.4 Summary

This chapter proposed a location-content-aware recommender system, LCARS, which provides a user with spatial item recommendations within the querying city based on the individual interests and the local preferences mined from the user's activity history. LCARS can facilitate people's travel not only in their home area but also in a new city where they have no activity history. By taking advantage of both the content and location information of spatial items, our system overcomes the data sparsity problem in the original user-item matrix. We evaluated our system using extensive experiments based on two real datasets. According to the experimental results, our approach significantly outperforms existing recommendation methods in effectiveness. The results also justify each component proposed in our system, such as taking local preferences and item content information into account, and the proposed scalable query processing technique improves the efficiency of our approach significantly.

References

1. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans. Knowl. Data Eng. **17**(6), 734–749 (2005). June
2. Bao, J., Zheng, Y., Mokbel, M.F.: Location-based and preference-aware recommendation using sparse geo-social networking data. In: SIGSPATIAL, pp. 199–208 (2012)
3. Chen, W.-Y., Chu, J.-C., Luan, J., Bai, H., Wang, Y., Chang, E.Y.: Collaborative filtering for orkut communities: discovery of user latent behavior. In: WWW, pp. 681–690 (2009)

4. Cho, E., Myers, S.A., Leskovec, J.: Friendship and mobility: user movement in location-based social networks. In: KDD, pp. 1082–1090 (2011)
5. Cremonesi, P., Koren, Y., Turrin, R.: Performance of recommender algorithms on top-n recommendation tasks. In: RecSys, pp. 39–46 (2010)
6. Gao, H., Tang, J., Liu, H.: gSCorr: modeling geo-social correlations for new check-ins on location-based social networks. In: CIKM, pp. 1582–1586 (2012)
7. Horozov, T., Narasimhan, N., Vasudevan, V.: Using location for personalized poi recommendations in mobile environments. In: SAINT, pp. 124–129 (2006)
8. Jin, X., Zhou, Y., Mobasher, B.: A maximum entropy web recommendation system: combining collaborative and content features. In: KDD, pp. 612–617 (2005)
9. Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: KDD, pp. 426–434 (2008)
10. Levandoski, J.J., Sarwat, M., Eldawy, A., Mokbel, M.F.: Lars: A location-aware recommender system. In: ICDE, pp. 450–461 (2012)
11. Linden, G., Smith, B., York, J.: Amazon.com recommendations: Item-to-item collaborative filtering. IEEE Internet Comput. **7**(1), 76–80 (2003)
12. Liu, X., He, Q., Tian, Y., Lee, W.-C., McPherson, J., Han, J.: Event-based social networks: linking the online and offline social worlds. In: KDD, pp. 1032–1040 (2012)
13. Ma, H., King, I., Lyu, M.R.: Learning to recommend with social trust ensemble. In: SIGIR, pp. 203–210 (2009)
14. Ricci, F., Rokach, L., Shapira, B., Kantor, P.B.: Recommender Systems Handbook. Springer-Verlag New York Inc, New York (2010)
15. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: WWW, pp. 285–295 (2001)
16. Scellato, S., Noulas, A., Lambiotte, R., Mascolo, C.: Socio-spatial properties of online location-based social networks. In: ICWSM (2011)
17. Tang, J., Wu, S., Sun, J., Su, H.: Cross-domain collaboration recommendation. In: KDD, pp. 1285–1293 (2012)
18. Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., Su, Z.: Arnetminer: extraction and mining of academic social networks. In: KDD, pp. 990–998 (2008)
19. Ye, M., Yin, P., Lee, W.-C., Lee, D.-L.: Exploiting geographical influence for collaborative point-of-interest recommendation. In: SIGIR, pp. 325–334 (2011)
20. Yin, H., Cui, B., Li, J., Yao, J., Chen, C.: Challenging the long tail recommendation. Proc. VLDB Endow. **5**(9), 896–907 (2012)
21. Yin, H., Cui, B., Sun, Y., Hu, Z., Chen, L.: Lcars: A spatial item recommender system. ACM Trans. Inf. Syst. **32**(3), 11:1–11:37 (2014)
22. Yin, H., Sun, Y., Cui, B., Hu, Z., Chen, L.: Lcars: A location-content-aware recommender system. In: KDD, pp. 221–229 (2013)

Chapter 4
Location-Based and Real-Time Recommendation

Abstract 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, extreme sparsity of user-POI matrix creates a severe challenge. To cope with this challenge, we propose a unified probabilistic generative model, *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, i.e., hometown recommendation and out-of-town recommendation. TRM effectively overcomes the 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 recommendation. To support real-time POI recommendation, we further extend the TRM model to an online learning model TRM-Online to track changing user interests and speed up the model training. We conduct extensive experiments to evaluate the performance of our proposals on two real-world datasets including recommendation effectiveness, overcoming cold-start problem and model training efficiency. The experimental results demonstrate the superiority of our TRM models, especially the TRM-Online, compared with the state-of-the-art competitive methods, by making more effective and efficient mobile recommendations. Besides, we study the importance of each type of patterns 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.

Keywords Point of interest · Real-time recommendation · Online learning · Dynamic user interest modeling

H. Yin and B. Cui, *Spatio-Temporal Recommendation in Social Media*,
SpringerBriefs in Computer Science, DOI 10.1007/978-981-10-0748-4_4

4.1 Introduction

With the rapid development of Web 2.0, 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 [12]. 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 a user travels to an unfamiliar area, where she has little knowledge about the neighborhood. In this scenario, the recommender system is proposed as *recommendation for out-of-town users* in [12]. In this chapter, 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 extreme sparsity of 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 [18] made on LBSNs shows that most of 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 non-home cities are very few and only take up 0.47 % of the check-in records left in their home cities [25]. This observation of travel locality is quite common in the real world [25], aggravating the data sparsity problem with POI recommendation for out-of-town users (e.g., if we want to recommend POIs located at Los Angeles to people from New York City) [12, 32].

The most popular approach in recommender systems is the *collaborative filtering* [1]. There exists a considerable body of research [12, 14, 18, 20, 29] which deposited people's check-in history into user-POI matrix where 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 [12, 18, 29] 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 recommendation. 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 the similar users, most of the recommended POIs would be located in the target user's home town. So, these CF-based methods cannot be directly applied to the POI recommendation for out-of-town users [12, 32]. Besides, some recent

literatures [14, 20] adopted latent factor models (e.g., matrix factorization) to a user-POI matrix to alleviate the 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 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.

4.1.1 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 [35], namely 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 [30]. 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 of users have very small entropies. (2) Temporal Cyclic Patterns. As observed in [14, 38], 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 [31]. So, POIs visited by users often form spatial clusters, i.e., people tend to check in around several centers (e.g., "home" and "office") [7, 20]. Note that while there are some recent studies [7, 14, 20, 32, 38] that exploited one of the above 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 4.1, 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.

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

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

Table 4.1 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

as a distribution over topics. Specifically, we infer an individual's interests according to the semantic contents of her checked-in POIs, respectively. Thus, our model alleviates the data sparsity to some extent, especially for out-of-town recommendation scenario, as the semantic contents play the role of medium which can transfer users' interests inferred at their home regions to out-of-town regions where they are traveling. As the quality of the learnt 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 time are more likely to be restaurants, while the ones visited around midnight are more likely to be nightlife spots.

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 her historical visited POIs or her 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, i.e., the wisdom of crowds, 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., hometown) cannot always be applied to recommendation at another region. For example, a user u never goes gambling when he lives in Beijing, China, but when he travels to Macao or Las Vegas he 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.

4.1.2 Real-Time POI Recommendation

As the 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 model developed in [35] is incapable of supporting real-time

recommendation due to the following reasons. First, although TRM exploits the temporal cyclic effect of the public visiting POIs to improve the 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 [34]. 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 proved to be commercially very valuable since it indicates visiting and purchasing intents. Second, it is difficult to apply the current TRM model for large-scale check-in data which arrives in a stream, since the batch learning algorithm developed for the TRM in [35] needs to run through all check-in data for many times (about 1000 iterations), and it is very time-consuming and infeasible. Therefore, to support real-time POI recommendation, we extend the batch TRM [35] to an online learning model TRM-Online in this article, which can efficiently process the check-in stream and track changing user interests.

To demonstrate the applicability of TRM models, we investigate how they support two recommendation scenarios in a unified way: (1) hometown recommendation which assumes that the target user is located in her hometown, i.e., to meet users' information needs in their daily life, and (2) out-of-town recommendation that aims to cater to users when they travel out of town, especially in an unfamiliar region. The real-time recommendation requires that both of the recommendation scenarios should be time-aware [38], location-based [12] and personalized, i.e., to recommend different ranked lists of POIs for the same target user at different time 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 naive approach to produce online top-k recommendations is to first compute a ranking score for each POIs and then select k ones with highest ranking scores.

The remainder of the chapter is organized as follows. Section 4.2 details TRM model and its batch learning algorithm. We present an online learning algorithm for the TRM in Sect. 4.3. We present how to deploy TRM models to two typical POI recommendation scenarios and overcome the cold-start problem in Sect. 4.4. We describe the experimental setup and report the experimental results in Sect. 4.5. We conclude the chapter in Sect. 4.6.

4.2 Joint Modeling of User Check-In Activities

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

4.2.1 Preliminary

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

Definition 4.1 (*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. Besides, there is textual semantic information associated with a POI, such as the category and tag words. We use the notation W_v to denote the set of words describing POI v.

Definition 4.2 (*User Home Location*) Following the recent work of [19], 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 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 her (e.g., the places where she 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 [25]. This method discretizes the world into 25 km-by-25 km cells and finds the one with most of her check-ins. Then, her home location is defined as the average position of all her check-ins within the cell.

Definition 4.3 (*Check-in Activity*) A check-in activity is represented by a five tuple (u, v, l_v, W_v, t) that means user u visits POI v at time t.

As suggested in [14, 38], human geographical movement exhibits significant temporal cyclic patterns on LBSNs which 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 chapter unless noted otherwise.

Definition 4.4 (*User Document*) For each user u, we create a user document D_u, which is a set of check-in activities associated with user u. The dataset D used in our model consists of user documents, i.e., $D = \{D_u : u \in U\}$ where U is the set of users.

Definition 4.5 (*Topic*) Given a collection of words W, a topic z is defined as a multinomial distribution over W, i.e., $\phi_z = \{\phi_{z,w} : w \in W\}$ 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 D as the union of a collection of user documents, we aim to provide location-based and real-time recommendation for both hometown and out-of-town users. We formulate our problem that takes into account both of the two scenarios in a unified fashion as follows.

Problem 4.1 (*Location-based and Real-time Recommendation*) Given a user check-in activity dataset D and a target user u_q with her current location l_q and time t_q (that is, 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 her home location (that is, $|l_q - l_{u_q}|$) is greater than d. Otherwise, the problem is a **hometown recommendation**.

Following related studies [12, 24], we set $d = 100$ km in our work, since a distance around 100 km is the typical radius of human "reach"—it takes about 1–2 h to drive such a distance.

4.2.2 Model Structure

To model user check-in activities, we propose a unified probabilistic generative model TRM to simulate the process of user's decision-making for the selection of POIs. Figure 4.1 shows the graphical representation of TRM where 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 4.2. Our input data, i.e., users' check-in records, are modeled as observed random variables, shown as shaded circles in Fig. 4.1. 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,

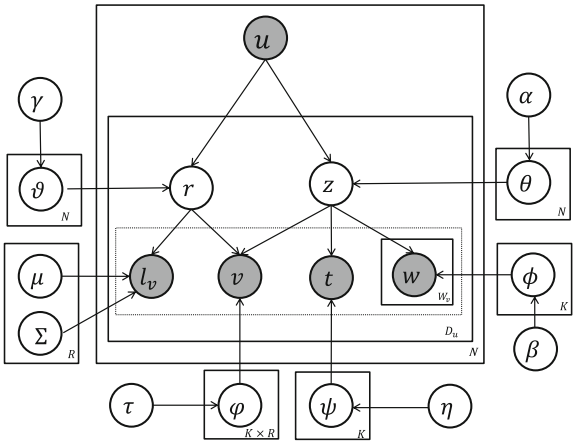

Fig. 4.1 The graphical representation of TRM

Table 4.2 Notations of parameters used in TRM

Variable	Interpretation
ϑ_u	The spatial patterns of user u, expressed by a multinomial distribution over a set of regions
θ_u	The interests of user u, expressed by a multinomial distribution over a set of topics
ϕ_z	A multinomial distribution over words specific to topic z
ψ_z	A multinomial distribution over time slots specific to topic z
$\varphi_{z,r}$	A multinomial distribution over POI IDs specific to topic-region (z, r)
μ_r	The mean location of region r
Σ_r	The location covariance of region r
$\gamma, \alpha, \beta, \eta, \tau$	Dirichlet priors to multinomial distributions ϑ_u, θ_u, ϕ_z, ψ_z and $\varphi_{z,r}$, respectively

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 her personal interests with the contents of that POI. Inspired by the early work about user interest modeling [16, 17, 23, 32], TRM adopts latent topics to characterize users' interests to overcome the data sparsity of user-word matrix. Specifically, we infer individual user's interest distribution over a set of topics according to the contents (e.g., tags and categories) of her checked-in POIs, denoted as θ_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, and 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 time 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 W_v and check-in time t. Thus, each topic z in TRM is not only associated with a word distribution ϕ_z, but also with a distribution over time ψ_z. This design enables ϕ_z and ψ_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 [14, 38] to split a day into hour-based slots.

In the standard topic models [4, 28], 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 short document W_v, it is most likely to be about a single topic. We therefore assign a single

topic to the document W_v. Similar idea of assigning a single topic to a twitter post has been used before [40].

User Mobility Modeling. Different from users' online behaviors in the virtual world, users' check-in activities in the physical world are limited by travel distance. So, 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 which are usually limited to some specific geographical regions [31] (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 literatures [21, 22], we assume a Gaussian distribution for each region r, and the location for POI v is characterized by $l_v \sim \mathcal{N}(\mu_r, \Sigma_r)$, as follows:

$$P(l_v|\mu_r, \Sigma_r) = \frac{1}{2\pi\sqrt{|\Sigma_r|}} \exp\left(\frac{-(l_v - \mu_r)^T \Sigma_r^{-1}(l_v - \mu_r)}{2}\right) \tag{4.1}$$

where μ_r and Σ_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 which receive few check-ins. We apply a multinomial distribution ϑ_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 which is responsible for generating the IDs of visited POIs, i.e., a topic-region (z, r) is associated with a distribution over POI IDs (that is $\varphi_{z,r}$). 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 like concert, ticket, bird, running, etc., 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 [37].

Algorithm 2: Probabilistic generative process in TRM

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

4.2.3 Generative Process

The generative process of TRM is summarized in Algorithm 2. To avoid overfitting, we place a Dirichlet prior [4, 28] over each multinomial distribution. Thus, θ_u, ϑ_u, ϕ_z, ψ_z and $\varphi_{z,r}$ are generated by Dirichlet distributions with parameters α, γ, β, η and τ, respectively.

Given a user u, when she plans to visit a POI v, she first selects a topic z according to her interest distribution θ_u. With the chosen topic z, words W_v are generated from the topic's word distribution ϕ_z, and time t is generated from the topic's temporal distribution ψ_z. Besides the topic, she also needs to choose a region r according to her spatial distribution ϑ_u. With the chosen region r, the POI's geographical coordinate l_v is generated by the region's spatial distribution $\mathcal{N}(\mu_r, \Sigma_r)$. Finally, with the chosen topic z and region r, the POI indicator v is generated by the joint topic-region factor $\varphi_{z,r}$ which is a multinomial distribution over POI IDs.

4.2.4 Model Inference

Our goal is to learn parameters that maximize the marginal log-likelihood of the observed random variables $\boldsymbol{v}, \boldsymbol{l}_v, \boldsymbol{W}_v$ and $\boldsymbol{t}$. However, the exact marginalization is difficult due to the intractable normalizing constant of the posterior distribution. Therefore, we follow the studies [27, 32] to use collapsed Gibbs sampling for approximate inference, i.e., to maximize the complete data likelihood in Eq. (4.2). As a widely used Markov chain Monte Carlo (MCMC) algorithm, Gibbs Sampling iteratively samples latent variables (i.e., $\{z,\ r\}$ 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., $\{\theta, \vartheta, \phi, \psi, \varphi\}$). For simplicity and speed, we estimate the Gaussian distribution parameters (μ_r, Σ_r) by the method of moments. As for the hyperparameters $\alpha, \beta, \gamma, \eta$ and τ, for simplicity, we take fixed values, i.e., $\alpha = 50/K$, $\gamma = 50/R$ and $\beta = \eta = \tau = 0.01$, following the studies [27, 32]. 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 region r for each check-in record $(u,\ v,\ l_v,\ W_v,\ t)$, i.e., we need to compute the conditional probabilities $P(z|\boldsymbol{z}_\neg, \boldsymbol{r}, \boldsymbol{v}, \boldsymbol{l}_v, \boldsymbol{W}_v, \boldsymbol{t}, \boldsymbol{u}, \cdot)$ and $P(r|\boldsymbol{z}, \boldsymbol{r}_\neg, \boldsymbol{v}, \boldsymbol{l}_v, \boldsymbol{W}_v, \boldsymbol{t}, \boldsymbol{u}, \cdot)$, where $\boldsymbol{z}_\neg$ and $\boldsymbol{r}_\neg$ represents 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 Eq. (4.2).

$$
\begin{aligned}
&P(\boldsymbol{v}, \boldsymbol{l}_v, \boldsymbol{W}_v, \boldsymbol{t}, \boldsymbol{z}, \boldsymbol{r}|\alpha, \beta, \gamma, \tau, \eta, \mu, \Sigma) \\
&\quad = P(\boldsymbol{z}|\alpha)P(\boldsymbol{W}_v|\boldsymbol{z}, \beta)P(\boldsymbol{t}|\boldsymbol{z}, \eta)P(\boldsymbol{r}|\gamma)P(\boldsymbol{v}|\boldsymbol{z}, \boldsymbol{r}, \tau)P(\boldsymbol{l}_v|\boldsymbol{r}, \mu, \Sigma) \\
&\quad = P(\boldsymbol{z}|\alpha) \prod_{\boldsymbol{w} \in \boldsymbol{W}_v} P(\boldsymbol{w}|\boldsymbol{z}, \beta)P(\boldsymbol{t}|\boldsymbol{z}, \eta)P(\boldsymbol{r}|\gamma)P(\boldsymbol{v}|\boldsymbol{z}, \boldsymbol{r}, \tau)P(\boldsymbol{l}_v|\boldsymbol{r}, \mu, \Sigma) \\
&\quad = \int P(\boldsymbol{z}|\theta)P(\theta|\alpha)d\theta \prod_{\boldsymbol{w} \in \boldsymbol{W}_v} \int P(\boldsymbol{w}|\boldsymbol{z}, \phi)P(\phi|\beta)d\phi \int P(\boldsymbol{t}|\boldsymbol{z}, \psi)P(\psi|\eta)d\psi \\
&\qquad \times \int P(\boldsymbol{r}|\vartheta)P(\vartheta|\gamma)d\vartheta \int P(\boldsymbol{v}|\boldsymbol{z}, \boldsymbol{r}, \varphi)P(\varphi|\tau)d\varphi P(\boldsymbol{l}_v|\boldsymbol{r}, \mu, \Sigma)
\end{aligned} \tag{4.2}
$$

Sampling topic indicator z for check-in $(u,\ v,\ l_v,\ W_v,\ t)$ according to:

$$
\begin{aligned}
P(z|\boldsymbol{z}_\neg, \boldsymbol{r}, \boldsymbol{v}, \boldsymbol{l}_v, \boldsymbol{W}_v, \boldsymbol{t}, \boldsymbol{u}, \cdot) &\propto \frac{n_{u,z,\neg} + \alpha}{\sum_{z'}(n_{u,z',\neg} + \alpha)} \\
&\times \frac{n_{z,t,\neg} + \eta}{\sum_{t'}(n_{z,t',\neg} + \eta)} \frac{n_{z,r,v,\neg} + \tau}{\sum_{v'}(n_{z,r,v',\neg} + \tau)} \prod_{w \in W_v} \frac{n_{z,w,\neg} + \beta}{\sum_{w'}(n_{z,w',\neg} + \beta)}
\end{aligned} \tag{4.3}
$$

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; $n_{z,r,v}$ 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 r for check-in $(u,\ v,\ l_v,\ W_v,\ t)$ according to:

$$\begin{aligned} &P(r|\boldsymbol{r}_{\neg}, \boldsymbol{z}, \boldsymbol{v}, \boldsymbol{l_v}, \boldsymbol{W_v}, \boldsymbol{t}, \boldsymbol{u}, \cdot) \\ &\propto \frac{n_{u,r,\neg}+\gamma}{\sum_{r'}(n_{u,r',\neg}+\gamma)} \frac{n_{z,r,v,\neg}+\tau}{\sum_{v'}(n_{z,r,v',\neg}+\tau)} P(l_v|\mu_r,\ \Sigma_r) \end{aligned} \tag{4.4}$$

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., μ and Σ) according to the assigned regions $\boldsymbol{r}$ for simplicity and speed. Specifically, parameters μ_r and Σ_r are updated as in Eqs. (4.5) and (4.6).

$$\mu_r = E(r) = \frac{1}{|C_r|} \sum_{v \in C_r} l_v \tag{4.5}$$

$$\Sigma_r = D(r) = \frac{1}{|C_r| - 1} \sum_{v \in C_r} (l_v - \mu_r)(l_v - \mu_r)^T \tag{4.6}$$

where C_r denotes the collection of POIs which 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 (μ and Σ) 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), and then randomly initialize the topic assignments for the check-in records (Lines 5–9). Afterwards, in each iteration, Eqs. (4.3), and (4.4) are utilized to update the region and topic assignments for each check-in record $(u,\ v,\ l_v,\ W_v,\ t)$ (Lines 14–15). After each sampling, we update the Gaussian distribution parameters (Lines 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 the above inference framework as follows. Suppose the process needs I iterations to reach convergence. In each iteration, it requires to go through all user check-in records. For each check-in record, it first requires $O(K)$ operations to compute the posterior distribution for sampling latent topic, and then needs $O(R)$ operations to compute the posterior distribution for sampling latent region. Thus, the whole time complexity is $O(I(K + R)|D|)$, which is linear to the size of the dataset (i.e., the number of check-ins D).

4.3 Online Learning for TRM

In practice, users' check-in records are generated in a real-time manner and continually arrive in the form of data stream. The batch learning algorithms such as our developed Algorithm 3 usually suffer from the following two drawbacks when dealing with the situation mentioned above: (1) Delay on model updates caused by the expensive time cost of re-running the batch model; and (2) Disability to track changing user interests and spatial mobility patterns due to the fact that 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 recommendation, we develop an online learning model TRM-Online using particle filter.

4.3.1 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, W_v, t) as an observation variable. Particle Filter is a kind of efficient Markov Chain Monte Carlo (MCMC) sampling method for estimating parameters of state space model. Being different from Gibbs sampling, it is an online algorithm.

Particle filter can be applied to solve the inference of TRM based on the following two reasons. First, suppose x is the hidden state variable and y is the observation variable in the particle filter. The objective of 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 TRM is to estimate the posterior distribution $P(\boldsymbol{r}, \boldsymbol{z}|\boldsymbol{v}, \boldsymbol{l_v}, \boldsymbol{W_v}, \boldsymbol{t}, \cdot)$, which is also the probability of state variables given observations. Second, 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, RTM meets this requirement.

In this chapter, we employ Rao–Blackwellized particle filtering (RBPF), an enhanced version of particle filtering [10] as our method. RBPF integrates out the latent variables, and thus makes the solutions simpler. In our algorithm, we have P particles, and each particle p represents an answer that we desire, i.e., 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. What is more, a reassignment process is added to enhance the quality of samples.

Algorithm 3: Inference Framework of TRM

Input: user check-in collection D, number of iteration I, number of burnin I_b, sample lag I_s, Priors α, γ, β, η, τ

Output: estimated parameters $\hat{\theta}$, $\hat{\vartheta}$, $\hat{\phi}$, $\hat{\varphi}$, $\hat{\psi}$, $\hat{\mu}$, and $\hat{\Sigma}$

Create variables θ^{sum}, ϑ^{sum}, ϕ^{sum}, φ^{sum}, ψ^{sum}, μ^{sum} and Σ^{sum}, and initialize them with zero;
Create variables μ and Σ;
Initialize the clustering of geographical locations using K-Means method.
Update μ and Σ according to Eqs. (4.5) and (4.6), respectively;
for *each* $D_u \in D$ **do**
  for *each check-in record* $(u,\ v,\ l_v,\ W_v,\ t) \in D_u$ **do**
    Assign topic randomly;
  end
end
Initialize variable *count* with zero;
for *iteration* $= 1$ *to* I **do**
  for *each* $D_u \in D$ **do**
    for *each check-in record* $(u,\ v,\ l_v,\ W_v,\ t) \in D_u$ **do**
      Update topic assignment using Eq. (4.3);
      Update region assignment using Eq. (4.4);
      Update μ and Σ according to Eqs. (4.5) and (4.6), respectively;
    end
  end
  if *(iteration* $> I_b$*) and (iteration mod* $I_s == 0$*)* **then**
    $count = count + 1$;
    Update θ^{sum}, ϑ^{sum}, ϕ^{sum}, φ^{sum}, ψ^{sum}, μ^{sum} and Σ^{sum} as follows:

$$\theta^{sum}_{u,z} += \frac{n_{u,z} + \alpha}{\sum_{z'}(n_{u,z'} + \alpha)}$$

$$\vartheta^{sum}_{u,r} += \frac{n_{u,r} + \gamma}{\sum_{r'}(n_{u,r'} + \gamma)}$$

$$\phi^{sum}_{z,w} += \frac{n_{z,w} + \beta}{\sum_{w'}(n_{z,w'} + \beta)}$$

$$\varphi^{sum}_{z,r,v} += \frac{n_{z,r,v} + \tau}{\sum_{v'}(n_{z,r,v'} + \tau)}$$

$$\psi^{sum}_{z,t} += \frac{n_{z,t} + \eta}{\sum_{t'}(n_{z,t'} + \eta)}$$

$$\mu^{sum}_{r} += \mu_r$$

$$\Sigma^{sum}_{r} += \Sigma_r$$

  end
end
Return model parameters $\hat{\theta} = \frac{\theta^{sum}}{count}$, $\hat{\vartheta} = \frac{\vartheta^{sum}}{count}$, $\hat{\phi} = \frac{\phi^{sum}}{count}$, $\hat{\varphi} = \frac{\varphi^{sum}}{count}$, $\hat{\psi} = \frac{\psi^{sum}}{count}$, $\hat{\mu} = \frac{\mu^{sum}}{count}$, and $\hat{\Sigma} = \frac{\Sigma^{sum}}{count}$;

4.3.2 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 time stamps 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 D^s to denote the check-in dataset generated by users at the sth epoch. As shown in Algorithm 4, the overall algorithm consists of two phases: **initialization phase** and **online phase**. 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 $\hat{\Psi}$ after processing these records.

In the *initialization phase* (Lines 1–12), for each particle, we apply TRM-Batch on an initial dataset 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 and 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.

$$
\begin{aligned}
&P(z^p|\boldsymbol{z}^p_{\neg}, \boldsymbol{r}^p, \boldsymbol{v}, \boldsymbol{l_v}, \boldsymbol{W_v}, \boldsymbol{t}, \boldsymbol{u}, \cdot) \\
&\propto \frac{m^{p,s}_{u,z} + n^{p,s}_{u,z,\neg} + \alpha}{\sum_{z'}(m^{p,s}_{u,z'} + n^{p,s}_{u,z',\neg} + \alpha)} \\
&\quad \times \frac{m^{p,s}_{z,t} + n^{p,s}_{z,t,\neg} + \eta}{\sum_{t'}(m^{p,s}_{z,t'} + n^{p,s}_{z,t',\neg} + \eta)} \frac{m^{p,s}_{z,r,v} + n^{p,s}_{z,r,v,\neg} + \tau}{\sum_{v'}(m^{p,s}_{z,r,v'} + n^{p,s}_{z,r,v',\neg} + \tau)} \prod_{w \in W_v} \frac{m^{p,s}_{z,w} + n^{p,s}_{z,w,\neg} + \beta}{\sum_{w'}(m^{p,s}_{z,w'} + n^{p,s}_{z,w',\neg} + \beta)}
\end{aligned} \tag{4.7}
$$

$$
\begin{aligned}
&P(r^p|\boldsymbol{r}^p_{\neg}, \boldsymbol{z}^p, \boldsymbol{v}, \boldsymbol{l_v}, \boldsymbol{W_v}, \boldsymbol{t}, \boldsymbol{u}, \cdot) \\
&\propto \frac{m^{p,s}_{u,r} + n^{p,s}_{u,r,\neg} + \gamma}{\sum_{r'}(m^{p,s}_{u,r'} + n^{p,s}_{u,r',\neg} + \gamma)} \frac{m^{p,s}_{z,r,v} + n^{p,s}_{z,r,v,\neg} + \tau}{\sum_{v'}(m^{p,s}_{z,r,v'} + n^{p,s}_{z,r,v',\neg} + \tau)} P(l_v|\mu_r, \Sigma_r)
\end{aligned} \tag{4.8}
$$

where $\neg$ has a different meaning from that in Eqs. (4.3) and (4.4). In Eqs. (4.3) and (4.4), 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 the above equations includes two parts. The first part is the contribution of the previous epochs before 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, $n^{p,s}_{u,z}$ is the number of times that topic z is assigned to user u by the particle p at the sth epoch.

Algorithm 4: Online Inference of TRM

Input: user check-in collection $D = \bigcup_s D^s$;
Output: estimated parameter set $\hat{\Psi} = \{\hat{\theta}, \hat{\vartheta}, \hat{\phi}, \hat{\varphi}, \hat{\psi}, \hat{\mu}, \hat{\Sigma}\}$;

/* **Initialization Phase:** */
for $p = 1$ *to* P **do**
  while *burn-in point is not reached* **do**
    for *each* $D_u \in D^0$ **do**
      for *each check-in record* $(u,\ v,\ l_v,\ W_v,\ t) \in D_u$ **do**
        Draw topic sample z using Eq. (4.3);
        Draw region sample r using Eq. (4.4);
        Update μ^p and Σ^p using Eqs. (4.5) and (4.6), respectively;
      end
    end
  end
end
Calculate sufficient statistics for each particle p;
/* **Online Phase:** */
Initialize importance weights $\omega^p = 1/P$ for any $p \in \{1, \ldots, P\}$;
for $s = 1$ *to* S **do**
  for *each* $D_u \in D^s$ **do**
    for *each check-in record* $(u,\ v,\ l_v,\ W_v,\ t) \in D_u$ **do**
      for $p = 1$ *to* P **do**
        Draw topic sample z^p using Eq. (4.7);
        Draw region sample r^p using Eq. (4.8);
        Update μ^p and Σ^p using Eqs. (4.10–4.13), respectively;
        $\omega^p = \omega^p P(v,\ l_v,\ W_v,\ t | \boldsymbol{z}_{\neg}^p, \boldsymbol{r}_{\neg}^p, \boldsymbol{v}, \boldsymbol{l_v}, \boldsymbol{W_v}, \boldsymbol{t}, \boldsymbol{u}, \cdot)$;
      end
      Normalize ω^p for any $p \in \{1, \ldots, P\}$ as $\omega^p = \frac{\omega^p}{\sum_{p'} \omega^{p'}}$;
      Calculate N_{eff} using Eq. (4.15);
      if $N_{eff} \leq N_{thresh}$ **then**
        Sampling Importance Resample process;
        Randomly select a collection of check-in records $D(i)$;
        for *each check-in* $(u,\ v,\ l_v,\ W_v,\ t) \in D(i)$ **do**
          for $p = 1$ *to* P **do**
            Draw topic sample z^p using Eq. (4.7);
            Draw region sample r^p using Eq. (4.8);
            Update μ^p and Σ^p using Eqs. (4.10)–(4.13), respectively;
          end
        end
        for $p = 1$ *to* P **do**
          set $\omega^p = 1/P$;
        end
      end
    end
  end
end
Generate parameter set $\hat{\Psi}$;

Following the recent work [2], we use exponential decay with kernel parameter κ defined as follows:

$$m^{p,s} = \sum_{h=0}^{s-1} \exp\left(\frac{h-s}{\kappa}\right) n^{p,h}, \tag{4.9}$$

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 Eqs. (4.5) and (4.6), we update the model parameters μ_r^p and Σ_r^p, as follows:

$$\mu_r^{p,s} = \frac{1}{|C_r^{p,s}|} \sum_{v \in C_r^p} l_v \tag{4.10}$$

$$\Sigma_r^{p,s} = \frac{1}{|C_r^{p,s}| - 1} \sum_{v \in C_r^p} (l_v - \mu_r)(l_v - \mu_r)^T \tag{4.11}$$

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

$$\mu_r^p = \frac{\mu_r^{p,s} + \exp(\frac{-1}{\kappa})\mu_r^{p,s-1}}{1 + \exp(\frac{-1}{\kappa})} \tag{4.12}$$

$$\Sigma_r^p = \frac{\Sigma_r^{p,s} + \exp(\frac{-1}{\kappa})\Sigma_r^{p,s-1}}{1 + \exp(\frac{-1}{\kappa})} \tag{4.13}$$

In this algorithm, $P(z^p | \boldsymbol{z}_{\neg}^p, \boldsymbol{r}^p, \boldsymbol{v}, \boldsymbol{l_v}, \boldsymbol{W_v}, \boldsymbol{t}, \boldsymbol{u}, \cdot)$ and $P(r^p | \boldsymbol{r}_{\neg}^p, \boldsymbol{z}, \boldsymbol{v}, \boldsymbol{l_v}, \boldsymbol{W_v}, \boldsymbol{t}, \boldsymbol{u}, \cdot)$ in Eqs. (4.7) and (4.8) are selected as the proposal distributions, so the importance weights are updated as in Eq. (4.14), and then normalized to sum to 1. $P(v,\ l_v,\ W_v,\ t | \boldsymbol{z}_{\neg}^p, \boldsymbol{r}_{\neg}^p, \boldsymbol{v}, \boldsymbol{l_v}, \boldsymbol{W_v}, \boldsymbol{t}, \boldsymbol{u}, \cdot)$ is the probability of user u generating the current check-in record based on all sampled topic and region assignments so far.

$$\begin{aligned} \omega^p &= \omega^p P(v,\ l_v,\ W_v,\ t | \boldsymbol{z}_{\neg}^p, \boldsymbol{r}_{\neg}^p, \boldsymbol{v}, \boldsymbol{l_v}, \boldsymbol{W_v}, \boldsymbol{t}, \boldsymbol{u}, \cdot) \\ &= \omega^p \textstyle\sum_z \theta_{u,z}^p \psi_{z,t}^p \prod_{w \in W_v} \phi_{z,w}^p \sum_r \vartheta_{u,r}^p \varphi_{z,r,v}^p P(l_v | \mu_r^p,\ \Sigma_r^p) \end{aligned} \tag{4.14}$$

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:

$$\theta_{u,z}^p = \frac{m_{u,z}^{p,s} + n_{u,z,\neg}^{p,s} + \alpha}{\sum_{z'} (m_{u,z'}^{p,s} + n_{u,z',\neg}^{p,s} + \alpha)}$$

$$\vartheta_{u,r}^p = \frac{m_{u,r}^{p,s} + n_{u,r,\neg}^{p,s} + \gamma}{\sum_{r'} (m_{u,r'}^{p,s} + n_{u,r',\neg}^{p,s} + \gamma)}$$

$$\phi_{z,w}^{p} = \frac{m_{z,w}^{p,s} + n_{z,w,\neg}^{p,s} + \beta}{\sum_{w'}(m_{z,w'}^{p,s} + n_{z,w',\neg}^{p,s} + \beta)}$$

$$\psi_{z,t}^{p} = \frac{m_{z,t}^{p,s} + n_{z,t,\neg}^{p,s} + \eta}{\sum_{t'}(m_{z,t'}^{p,s} + n_{z,t',\neg}^{p,s} + \eta)}$$

$$\varphi_{z,r,v}^{p} = \frac{m_{z,r,v}^{p,s} + n_{z,r,v,\neg}^{p,s} + \tau}{\sum_{v'}(m_{z,r,v'}^{p,s} + n_{z,r,v',\neg}^{p,s} + \tau)}$$

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

$$N_{eff} = 1/\sum_{p=1}^{P}(\omega^{p})^{2} \tag{4.15}$$

N_{eff} measures the efficiency of the method and controls the algorithm to avoid degeneracy [11]. 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 high possibility to be eliminated. This process reflects the "survival of the fittest" law, i.e., "excellent" solutions should be inherited. Intuitively, N_{thresh} decides the frequency of resample, and thus influences the effectiveness and speed of the algorithm.

In addition, we add a re-assignment process (Lines 28–34) to improve the quality of samples. Since check-ins coming in online phase are only sampled once, the result might be inaccurate. We solve this problem by picking up some check-ins randomly, and re-assigning topics and regions to them. D_i is a collection of randomly selected check-in records whose number is no more than i. For each check-in in D_i, we sample a new topic and a new region according to Eqs. (4.7) and (4.8). Obviously, when $|D_i|$ is big enough, TRM-Online will degenerate to a batch learning model, since previous check-ins will be re-assigned constantly.

Generally, our final objectives, the posterior distribution $P(\boldsymbol{r}, z|\boldsymbol{v}, \boldsymbol{l}_{\boldsymbol{v}}, \boldsymbol{W}_{\boldsymbol{v}}, \boldsymbol{t}, \cdot)$ as we mentioned in Sect. 4.3.1 is approximated as follows:

$$P(\boldsymbol{r}, z|\boldsymbol{v}, \boldsymbol{l}_{\boldsymbol{v}}, \boldsymbol{W}_{\boldsymbol{v}}, \boldsymbol{t}) \approx \sum_{p=1}^{P} \omega^{p} I(z, z^{p}) I(\boldsymbol{r}, \boldsymbol{r}^{p}) \tag{4.16}$$

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

$$I(z, z^{p}) = \begin{cases} 1 \text{ if } z \textit{ equals } z^{p} \\ 0 \quad \text{otherwise} \end{cases} \tag{4.17}$$

$$I(\boldsymbol{r}, \boldsymbol{r}^p) = \begin{cases} 1 \text{ if } \boldsymbol{r} \textit{ equals } \boldsymbol{r}^p \\ 0 \quad \text{otherwise} \end{cases} \tag{4.18}$$

In other words, particles with the same vectors $\boldsymbol{z}$ and $\boldsymbol{r}$ have the same $P(\boldsymbol{r}, \boldsymbol{z}|\boldsymbol{v}, \boldsymbol{l}_v, \boldsymbol{W}_v, \boldsymbol{t})$, 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:

$$\boldsymbol{z}^*, \boldsymbol{r}^* = \arg_{\boldsymbol{z}^p, \boldsymbol{r}^p} \max P(\boldsymbol{r}^p, \boldsymbol{z}^p | \boldsymbol{v}, \boldsymbol{l}_v, \boldsymbol{W}_v, \boldsymbol{t}) \tag{4.19}$$

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 same particles. Therefore, we modify this procedure to choose the particle with biggest weight for model parameter estimation. The modified $\boldsymbol{z}^*$ and $\boldsymbol{r}^*$ are as follows:

$$\boldsymbol{z}^* = \boldsymbol{z}^{p^*}, \quad \boldsymbol{r}^* = \boldsymbol{r}^{p^*}, \quad p^* = \arg_p \max \omega^p \tag{4.20}$$

With $\boldsymbol{z}^*$ and $\boldsymbol{r}^*$, we can easily estimate the model parameters $\hat{\Psi}$ 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 [2, 26]. 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 [26]. The later 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.

4.4 POI Recommendation Using TRM

Once we have learnt the model parameter set $\hat{\Psi} = \{\hat{\theta}, \hat{\vartheta}, \hat{\phi}, \hat{\varphi}, \hat{\psi}, \hat{\mu}, \hat{\Sigma}\}$, given a target user u_q with the current time t_q and location l_q, i.e., $q = (u_q, t_q, l_q)$, we compute a probability of user u_q checking-in each unvisited POI v as in Eq. (4.21), and then select top-k POIs with the highest probabilities for the target user.

$$\begin{aligned} P(v|q, \hat{\Psi}) &= \frac{P(v, t_q|u_q, l_q, \hat{\Psi})}{\sum_{v'} P(v', t_q)|u_q, l_q, \hat{\Psi})} \\ &\propto P(v, t_q|u_q, l_q, \hat{\Psi}) \end{aligned} \tag{4.21}$$

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

$$P(v, t_q|u_q, l_q, \hat{\Psi}) = \sum_r P(r|l_q, \hat{\Psi}) P(v, t_q|u_q, r, \hat{\Psi}) \tag{4.22}$$

where $P(r|l_q, \hat{\Psi})$ denotes the probability of user u_q lying in region r given her current location l_q, and it is computed as in Eq. (4.23) according to Bayes rule, in which the prior probability of latent region r can be estimated using Eq. (4.24), as follows.

$$P(r|l_q, \hat{\Psi}) = \frac{P(r)P(l_q|r, \hat{\Psi})}{\sum_{r'} P(r')P(l_q|r', \hat{\Psi})} \propto P(r)P(l_q|r, \hat{\Psi}) \tag{4.23}$$

$$P(r) = \sum_u P(r|u)P(u) = \sum_u \frac{N_u + \kappa}{\sum_{u'}(N_{u'} + \kappa)} \hat{\vartheta}_{u',r} \tag{4.24}$$

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 pseudo-count. Note that to support dynamic real-time recommendation, we compute the probability of u_q choosing region r according to her real-time location l_q instead of the spatial patterns (i.e., $\vartheta_{u,r}$) learnt from her historical check-in records, which distinguishes this work from the static recommendation scheme adopted by most POI recommendation work [14, 15, 17, 20, 31, 39].

$P(v, t_q|u_q, r, \hat{\Psi})$ is computed as in Eq. (4.25) where we adopt geometric mean for the probability of topic z generating word set W_v, i.e., $P(W_v|z, \hat{\Psi}) = \prod_{w \in W_v} P(w|z, \hat{\Psi})$, considering that the number of words associated with different POIs may be different.

$$\begin{aligned} P(v, t_q|u_q, r, \hat{\Psi}) &= P(l_v|r, \hat{\Psi}) \sum_z P(z|u_q, \hat{\Psi}) P(t_q|z, \hat{\Psi}) \\ &\times \left(\prod_{w \in W_v} P(w|z, \hat{\Psi}) \right)^{\frac{1}{|W_v|}} P(v|z, r, \hat{\Psi}) \end{aligned} \tag{4.25}$$

Based on Eqs. (4.22)–(4.25), the original Eq. (4.21) can be rewritten as in Eq. (4.26).

$$\begin{aligned} &P(v|u_q, t_q, l_q, \hat{\Psi}) \\ &\propto \sum_r \Bigg[P(r)P(l_q|r, \hat{\Psi})P(l_v|r, \hat{\Psi}) \sum_z P(z|u, \hat{\Psi})P(t_q|z, \hat{\Psi}) \\ &\quad \times \left(\prod_{w \in W_v} P(w|z, \hat{\Psi}) \right)^{\frac{1}{|W_v|}} P(v|z, r, \hat{\Psi}) \Bigg] \end{aligned}$$

$$= \sum_r \left[P(r)P(l_q|\hat{\mu}_r, \hat{\Sigma}_r)P(l_v|\hat{\mu}_r, \hat{\Sigma}_r) \sum_z \hat{\theta}_{u_q,z}\hat{\psi}_{z,t_q} \left(\prod_{w \in \mathscr{W}_v} \hat{\phi}_{z,w} \right)^{\frac{1}{|W_v|}} \hat{\varphi}_{z,r,v} \right]$$

$$= \sum_r \sum_z \left[P(r)P(l_q|\hat{\mu}_r, \hat{\Sigma}_r)P(l_v|\hat{\mu}_r, \hat{\Sigma}_r)\hat{\theta}_{u_q,z}\hat{\psi}_{z,t_q} \left(\prod_{w \in \mathscr{W}_v} \hat{\phi}_{z,w} \right)^{\frac{1}{|W_v|}} \hat{\varphi}_{z,r,v} \right] \tag{4.26}$$

$$\begin{aligned} P(v|u_q, t_q, l_q, \hat{\Psi}) &\propto \sum_r \left[P(r)P(l_q|r, \hat{\Psi})P(l_v|r, \hat{\Psi}) \right] \\ &\quad \sum_z \left[P(z|u_q, \hat{\Psi})P(t_q|z, \hat{\Psi}) \left(\prod_{w \in W_v} P(w|z, \hat{\Psi}) \right)^{\frac{1}{|W_v|}} \right] \\ &= \sum_r \left[P(r)P(l_q|\hat{\mu}_r, \hat{\Sigma}_r)P(l_v|\hat{\mu}_r, \hat{\Sigma}_r) \right] \\ &\quad \sum_z \left[\hat{\theta}_{u_q,z}\hat{\psi}_{z,t_q} \left(\prod_{w \in W_v} \hat{\phi}_{z,w} \right)^{\frac{1}{|W_v|}} \right] \end{aligned} \tag{4.27}$$

4.4.1 Fast Top-k Recommendation Framework

In this subsection, we propose an efficient ranking framework according to Eq. (4.26), as follows:

$$S(q, v) = \sum_{a=(z,r)} W(q, a)F(v, a) \tag{4.28}$$

$$W(q, a) = \hat{\theta}_{u_q,z}\hat{\psi}_{z,t_q}P(l_q|\hat{\mu}_r, \hat{\Sigma}_r) \tag{4.29}$$

$$F(v, a) = P(r)P(l_v|\hat{\mu}_r, \hat{\Sigma}_r)\hat{\varphi}_{z,r,v} \left(\prod_{w \in \mathscr{W}_v} \hat{\phi}_{z,w} \right)^{\frac{1}{|W_v|}} \tag{4.30}$$

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., $a = (z, r)$), and $W(q, a)$ represents the weight of query q on attribute a, and $F(v, a)$ represents the score of POI v with respect to attribute a. 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 $(\hat{\mu}_r, \hat{\Sigma}_r)$) 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.

4.4.2 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 cold-start recommendation scenario. For a cold-start POI v along with its location l_v and content words 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 cold-start POI v is computed still according to Eq. (4.21), but the probability $P(v, t_q|u_q, r, \hat{\Psi})$ is redefined as follows:

$$P(v, t_q|u_q, r, \hat{\Psi}) = P(l_v|r, \hat{\Psi}) \sum_z P(z|u_q, \hat{\Psi}) P(t_q|z, \hat{\Psi}) \Big(\prod_{w \in W_v} P(w|z, \hat{\Psi}) \Big)^{\frac{1}{|W_v|}}$$

Compared with Eq. (4.25), the probability $P(v|z, r, \hat{\Psi}) = \hat{\varphi}_{z,r,v}$ is not utilized in the above equation, since the ID of cold-start POI v is not available in the training dataset. Thus, for cold-start POI v, the original equation (4.21) can be reformulated as in Eq. (4.27) which shows that TRM can effectively alleviate the cold-start problem by leveraging the semantic, temporal and spatial patterns.

4.5 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 model training efficiency.

4.5.1 Datasets

Following the method developed in [13], we collected two large-scale real-life datasets to conduct experiments: Foursquare and Twitter. Since we focus on new

Table 4.3 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

POI recommendation, we removed the repeated check-ins from the two datasets. Their basic statistics are shown in Table 4.3.

Foursquare. This dataset contains the check-in history of 4,163 users who live in the California, USA. For each user, it contains her 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 relationship 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 Fig. 4.2a. 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 a sound proof of the significance of out-of-town recommendation.

Twitter. This dataset is based on the publicly available twitter dataset [8]. Twitter supports third-party location sharing services like Foursquare and Gowalla (where users of these services can share their check-ins on Twitter). But the original dataset does not contain the category and tag information about each POI. So, 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 above Foursquare dataset. But, this dataset does not contain user social network information. Figure 4.2b illustrates the distribution of the check-in activities across USA. 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 the above two datasets. Thus, we employ the widely adopted method in [9, 25, 36] 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 her check-ins. To make the experiments repeatable, we make the two datasets publicly available.[3]

[2] https://developer.foursquare.com/.

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

Fig. 4.2 Distribution of user check-in activities. **a** Foursquare dataset. **b** Twitter dataset

4.5.2 Comparative Approaches

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

SVDFeature. SVDFeature [6] 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 [33, 38] to segment them into bins and grid squares.

Category-based KNN (CKNN). CKNN, developed in [3], 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 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, and then recommendations are produced according to check-in records of these selected users.

LCA–LDA. LCA–LDA is a location-content-aware recommender model which is developed to support POI recommendation for users traveling in new cities [32]. 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 TRM, it does not consider the temporal and spatial patterns of users' check-in activities.

UPS-CF. UPS-CF, proposed in [12], is a collaborative recommendation framework which incorporates social influence to support out-of-town recommendation. This framework integrates user-based collaborative filtering and social-based collaborative filtering, i.e., to recommend POIs to a target user according to the check-in records of both her friends and similar users with her.

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

simplified version of TRM where we remove content information of check-in POIs, and the latent variable z only generates 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 only generates the word set W_v for each check-in record; and as the third simplified version, **TRM-S3** does not consider the geographical information of check-in POIs, and the latent variable r are thus removed.

4.5.3 Evaluation Methods

Recommendation Effectiveness To make the evaluation process fair and reproducible, we adopt the *methodological description framework* proposed in [5, 34] 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 D, or on each of the sub-datasets of D independently. We adopt the *user-centered base set condition*. Specifically, for each user u, her check-in records D_u are first divided into hometown check-ins D_u^{home} and out-of-town check-ins D_u^{out}, since our TRM is designed for both hometown and out-of-town recommendation, 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 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 4.3. Then, we perform the splitting independently on each user's hometown check-in document D_u^{home} and out-of-town check-in document 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 D_u^{home} and 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 hyper-parameters 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 which are geographically close to v are comparable with v. This design can effectively simulate the local competition effect and user behavior of choices.

The above condition combination is also called "uc_td_prop" for short. According to the above evaluation conditions, the whole dataset D is split into the training set D_{train} and the test set 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 $(u, v, l_v, W_v, t) \in D_{test}$, 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 [17, 32]. Specifically, for each check-in $(u,\ v,\ l_v,\ W_v,\ t)$ in D_{test}:

1. We compute the ranking score for POI v and all other POIs which 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 where 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:

$$\text{Accuracy@}k = \frac{\#hit@k}{|D_{test}|}$$

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

4.5.4 Recommendation Effectiveness

In this subsection, we report the comparison results between our proposed TRM models and other competitive methods with well-tuned parameters. Figures 4.3 and 4.4 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 where k is set to 1, 5, 10, 15, 20, as a greater value of k is usually ignored for a typical top-k recommendation task.

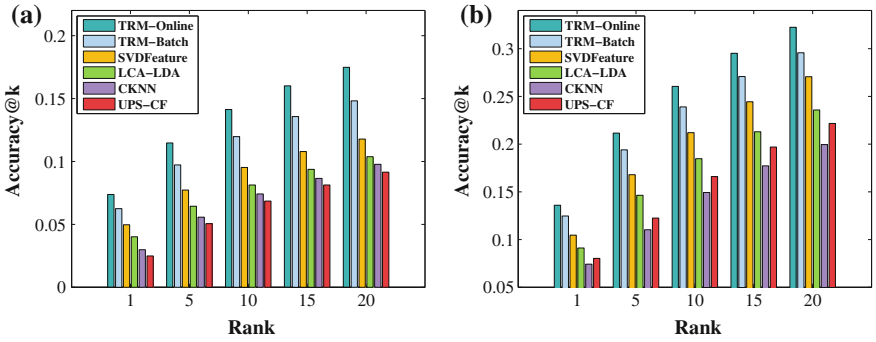

Fig. 4.3 Top-k performance on Foursquare dataset. **a** Out-of-town recommendation. **b** Hometown recommendation

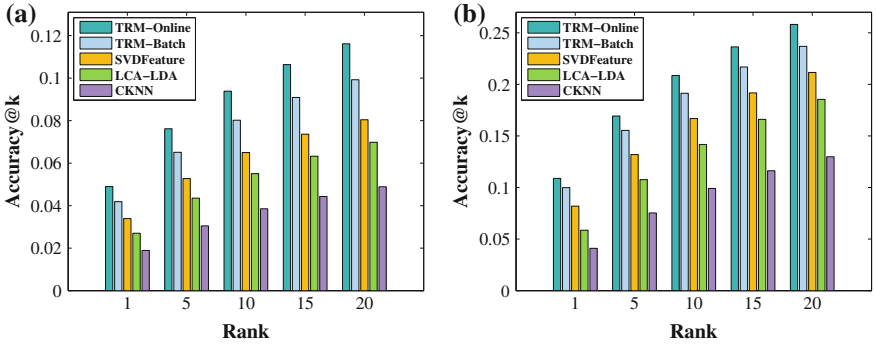

Fig. 4.4 Top-k performance on Twitter dataset. **a** Out-of-town recommendation. **b** Hometown recommendation

Out-of-Town Recommendation On Foursquare. Figure 4.3a presents the recommendation accuracy in the scenario of out-of-town recommendation, where 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 outperforms other competitive models significantly, and the advantages of TRM models over other competitive 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 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 SVD-Feature 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 θ_u are stable. That is to say, changes in user interests are not taken into consideration in TRM-Batch. However, users' interests are not always stable and may change as the 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 of China to Mountain View of the USA, she is most likely to change her interests. In our TRM-Online model, a POI that was visited recently by a user has a bigger impact on the prediction of her future visiting behaviors than an POI that was visited a long time ago.

Hometown Recommendation On Foursquare. In Fig. 4.3b, we report the performance of all recommendation models for the hometown scenario, and our TRM-Online achieves the highest recommendation accuracy. From the results, we observe that the recommendation accuracies of all methods are higher in Fig. 4.3b than that in Fig. 4.3a. Besides, CKNN outperforms UPS-CF in Fig. 4.3a while UPS-CF slightly exceeds CKNN in Fig. 4.3b, showing that the collaborative filtering better suits the hometown recommendation setting where 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, SVD-Feature and LCA–LDA are hybrid recommendation methods which take advantage of different dimension information, thus they perform well consistently in both recommendation settings. The comparison between Fig. 4.3a and Fig. 4.3b also reveals that the two recommendation scenarios are intrinsically different, and should be separately evaluated.

Recommendation on Twitter. Figure 4.4 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 Fig. 4.3, and 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.

4.5.5 Impact of Different Factors

In this subsection, 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, and the comparison results are shown in Tables 4.4 and 4.5. From the

Table 4.4 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

Table 4.5 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

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. Besides, 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 follows: Semantic Patterns > Temporal Patterns > Spatial Patterns, while in the hometown recommendation scenario they can be ranked as: Temporal Patterns > Spatial Patterns > Semantic Patterns. Obviously, 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 recommendation. This is because the two recommendation scenarios have different characteristics: (1) most of 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 hometown; and (3) users' daily routines may change when they travel out of town.

4.5.6 *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 Figs. 4.5 and 4.6. We define those POI 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 her check-in records associated with cold-start POIs as test data D_{test}, and the remaining check-in records as training data $D_{training}$. We aim to measure

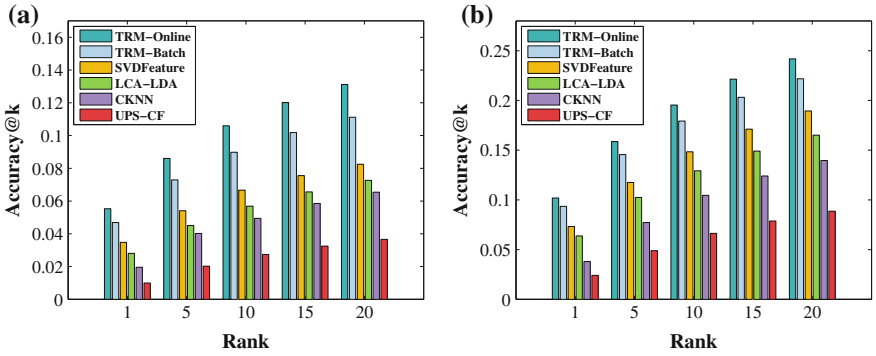

Fig. 4.5 Recommendation accuracy for cold-start POIs on Foursquare dataset. **a** Out-of-town recommendation. **b** Hometown recommendation

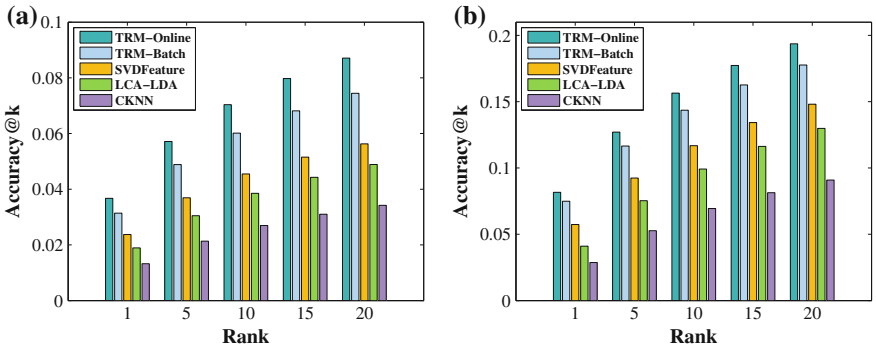

Fig. 4.6 Recommendation accuracy for cold-start POIs on Twitter dataset. **a** Out-of-town recommendation. **b** Hometown recommendation

whether the marked-off cold-start POIs in 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 Figs. 4.3 and 4.4, the recommendation accuracy of all algorithms decreases, to different degrees, for cold-start POIs, e.g., the recommendation accuracy of UPS-CF drops drastically while our TRM models deteriorate slightly; (3) all TRM, SVDFeature, LCA–LDA, 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 significant better than both collaborative filtering and social filtering methods. This is because cold-start POIs lacks 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 cold-start problem.

4.5.7 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 Fig. 4.7 shows, TRM-Batch costs much more time to get the new parameters compared with TRM-Online, especially when the number

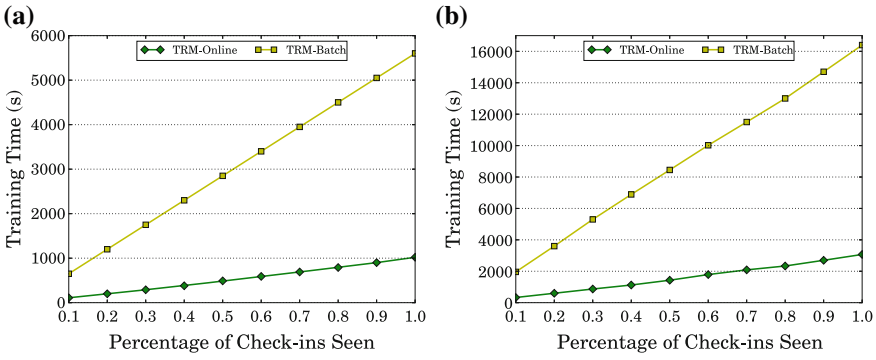

Fig. 4.7 Training time on two datasets. **a** On the Foursquare dataset. **b** On the Twitter dataset

of observed check-ins is large. When running on the Foursquare dataset, the training time of TRM-Online is less than 1000 s, while TRM-Batch takes more than 6,500 s, which is about 6.5 times longer. On the Twitter dataset, TRM-Batch takes more than 16,000 s, while TRM-Online takes less than 3,000 s. 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, and thus it is scalable to large-scale datasets.

In summary, besides the benefit of the higher recommendation accuracy as shown in Sect. 4.5.4, another advantage of TRM-Online is that it needs less training time.

4.6 Summary

In this article, we proposed a unified probabilistic generative 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 investigated how it supports two recommendation scenarios in a unified way, i.e., hometown recommendation and out-of-town recommendation. TRM can effectively overcome 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. 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, compared with the state-of-the-art competitors, by making more effective and efficient mobile recommendations. Besides, we studied the importance of each type of patterns in the two recommendation scenarios, respectively, and found that *semantic patterns* play a dominant role in overcoming the data sparsity in out-of-town recommendation, while *temporal patterns* are most important to improve hometown recommendation.

References

1. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans. Knowl. Data Eng. **17**(6), 734–749 (2005)
2. Ahmed, A., Low, Y., Aly, M., Josifovski, V., Smola, A.J.: Scalable distributed inference of dynamic user interests for behavioral targeting. In: KDD, pp. 114–122 (2011)
3. Bao, J., Zheng, Y., Mokbel, M.F.: Location-based and preference-aware recommendation using sparse geo-social networking data. In: SIGSPATIAL, pp. 199–208 (2012)
4. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. J. Mach. Learn. Res. **3**, 993–1022 (2003)

5. Campos, P.G., Díez, F., Cantador, I.: Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. User Model. User-Adapt. Interact. **24**(1–2), 67–119 (2014)
6. Chen, T., Zhang, W., Lu, Q., Chen, K., Zheng, Z., Svdfeature, YYu.: A toolkit for feature-based collaborative filtering. J. Mach. Learn. Res. **13**(1), 3619–3622 (2012)
7. Cheng, C., Yang, H., King, I., Lyu, M.R.: Fused matrix factorization with geographical and social influence in location-based social networks. In: AAAI (2012)
8. Cheng, Z., Caverlee, J., Lee, K., Sui, D.Z.: Exploring millions of footprints in location sharing services. In: ICWSM (2011)
9. Cho, E., Myers, S.A., Leskovec, J.: Friendship and mobility: user movement in location-based social networks. In: KDD, pp. 1082–1090 (2011)
10. Doucet, A. Freitas, N.d., Murphy, K.P., Russell, S.J. Rao-blackwellised particle filtering for dynamic bayesian networks. In: UAI, pp. 176–183 (2000)
11. Doucet, A., Godsill, S., Andrieu, C.: On sequential monte carlo sampling methods for bayesian filtering. Stat. Comput. **10**(3), 197–208 (2000)
12. Ference, G., Ye, M., Lee, W.C.: Location recommendation for out-of-town users in location-based social networks. In: CIKM, pp. 721–726 (2013)
13. Gao, H., Tang, J., Liu, H.: gscorr: modeling geo-social correlations for new check-ins on location-based social networks. In: CIKM, pp. 1582–1586 (2012)
14. Gao, H., Tang, J., Hu, X., Liu, H.: Exploring temporal effects for location recommendation on location-based social networks. In: RecSys, pp. 93–100 (2013)
15. Gao, H., Tang, J., Hu, X., Liu, H.: Content-aware point of interest recommendation on location-based social networks. In: AAAI (2015)
16. Hong, L., Ahmed, A., Gurumurthy, S., Smola, A.J., Tsioutsiouliklis, K.: Discovering geographical topics in the twitter stream. In: WWW, pp. 769–778 (2012)
17. Hu, B., Ester, M.: Spatial topic modeling in online social media for location recommendation. In: RecSys, pp. 25–32 (2013)
18. Levandoski, J.J., Sarwat, M., Eldawy, A., Mokbel, M.F.: Lars: a location-aware recommender system. In: ICDE, pp. 450–461 (2012)
19. Li, R., Wang, S., Deng, H., Wang, R., Chang, K.C.-C.: Towards social user profiling: unified and discriminative influence model for inferring home locations. In: KDD, pp. 1023–1031 (2012)
20. Lian, D., Zhao, C., Xie, X., Sun, G., Chen, E., Rui, Y.: Geomf: joint geographical modeling and matrix factorization for point-of-interest recommendation. In: KDD, pp. 831–840 (2014)
21. Lichman, M., Smyth, P.: Modeling human location data with mixtures of kernel densities. In: KDD, pp. 35–44 (2014)
22. Liu, B., Fu, Y., Yao, Z., Xiong, H.: Learning geographical preferences for point-of-interest recommendation. In: KDD, pp. 1043–1051 (2013)
23. Liu, B., Xiong, H.: Point-of-interest recommendation in location based social networks with topic and location awareness. In: SDM, pp. 396–404 (2013)
24. Mok, D., Wellman, B., Carrasco, J.: Does distance matter in the age of the internet? Urban Stud. **47**(13), 2747–2783 (2010)
25. Scellato, S., Noulas, A., Lambiotte, R., Mascolo, C.: Socio-spatial properties of online location-based social networks. In: ICWSM (2011)
26. Smola, A., Narayanamurthy, S.: An architecture for parallel topic models. Proc. VLDB Endow. **3**(1–2), 703–710 (2010)
27. Tang, J., Wu, S., Sun, J., Su, H.: Cross-domain collaboration recommendation. In: KDD, pp. 1285–1293 (2012)
28. Wallach, H.M., Mimno, D.M., McCallum, A.: Rethinking lda: why priors matter. In: NIPS, pp. 1973–1981 (2009)
29. Ye, M., Yin, P., Lee, W.-C.: Location recommendation for location-based social networks. In: GIS, pp. 458–461 (2010)
30. Ye, M., Shou, D., Lee, W.-C., Yin, P., Janowicz, K.: On the semantic annotation of places in location-based social networks. In: KDD, pp. 520–528 (2011)

31. Ye, M., Yin, P., Lee, W.-C., Lee, D.-L.: Exploiting geographical influence for collaborative point-of-interest recommendation. In: SIGIR, pp. 325–334 (2011)
32. Yin, H., Cui, B., Sun, Y., Hu, Z., Chen, L.: Lcars: a spatial item recommender system. ACM Trans. Inf. Syst., **32**(3), 11:1–11:37 (2014)
33. Yin, H., Cui, B., Chen, L., Hu, Z., Zhang, C.: Modeling location-based user rating profiles for personalized recommendation. ACM Trans. Knowl. Discov. Data **9**(3), 19:1–19:41 (2015)
34. Yin, H., Cui, B., Chen, L., Hu, Z., Zhou, X.: Dynamic user modeling in social media systems. ACM Trans. Inf. Syst. **33**(3), 10:1–10:44 (2015)
35. Yin, H., Cui, B., Huang, Z., Wang, W., Wu, X., Zhou, X.: Joint modeling of users' interests and mobility patterns for point-of-interest recommendation. In: MM, pp. 819–822 (2015)
36. Yin, H., Zhou, X., Shao, Y., Wang, H., Sadiq, S.: Joint modeling of user check-in behaviors for point-of-interest recommendation. In: CIKM, pp. 1631–1640 (2015)
37. Yin, Z., Cao, L., Han, J., Zhai, C., Huang, T.: Geographical topic discovery and comparison. In: WWW, pp. 247–256 (2011)
38. Yuan, Q., Cong, G., Ma, Z., Sun, A., Thalmann, N.M.: Time-aware point-of-interest recommendation. In: SIGIR, pp. 363–372 (2013)
39. Zhao, K., Cong, G., Yuan, Q., Zhu, K.: Sar: a sentiment-aspect-region model for user preference analysis in geo-tagged reviews. In: ICDE (2015)
40. Zhao, W.X., Jiang, J., Weng, J., He, J., Lim, E.-P., Yan, H., Li, X.: Comparing twitter and traditional media using topic models. In: ECIR, pp. 338–349 (2011)

Chapter 5
Fast Online Recommendation

Abstract Based on the spatiotemporal recommender models developed in the previous chapters, the top-k recommendation task can be reduced to an simple task of finding the top-k items with the maximum dot-products for the query/user vector over the set of item vectors. In this chapter, we build effective multidimensional index structures metric-tree and Inverted Index to manage the item vectors, and present three efficient top-k retrieval algorithms to speed up the online spatiotemporal recommendation. These three algorithms are metric-tree-based search algorithm (MT), threshold-based algorithm (TA), and attribute pruning-based algorithm (AP). MT and TA focus on pruning item search space, while AP aims to prune attribute space. To evaluate the performance of the developed techniques, we conduct extensive experiments on both real-world and large-scale synthetic datasets. The experimental results show that MT, TA, and AP can achieve superior performance under different data dimensionality.

Keywords Recommendation efficiency · Indexing structure · Top-k query processing · Metric tree · TA

5.1 Introduction

The spatiotemporal recommender systems based on latent-class models such as topic models and matrix factorization have demonstrated better accuracy than other methods such as nearest-neighbor models and graph-based models in the previous chapters. However, the online computational cost of finding the top-ranked items for every query/user in the system, once the models have been trained, has been rarely discussed in the academic literature.

In the latent-class models, the predicted ranking score of an item w.r.t. a query/use can often boil down to a dot-product between two vectors representing the query/user and the item, as shown in the previous three chapters. Given a query/user, the straightforward method of generating top-k recommendation needs to compute the ranking scores for all items and select the k ones with highest ranking scores, which is computationally inefficient, especially when the number of items or items' latent attributes

H. Yin and B. Cui, *Spatio-Temporal Recommendation in Social Media*,
SpringerBriefs in Computer Science, DOI 10.1007/978-981-10-0748-4_5

is large. Moreover, constructing the entire $\#QUERIES \times \#ITEMS$ preference matrix requires heavy computational (and space) resources. For example, the recently published Yelp's Challenge Dataset,[1] which contains 366,000 users and 61,000 business items. Generating the optimal recommendations in this dataset requires over 2.2×10^{11} dot-products using a naive algorithm. A 100-dimensional model required 56 h to find optimal recommendations for all the queries. In terms of storage, saving the whole preference matrix requires over 1TB of disk-space. Moreover, this dataset is just a small sample of the actual Yelp dataset and the problem worsens with larger numbers.

The online top-k recommendation can be formulated as: given a query q, we aim to find k top-ranked items with highest ranking scores over a set of items V, and the ranking score is computed as follows:

$$S(q, v) = \sum_a W(q, a) F(v, a) = \mathbf{q}^T \mathbf{v} = \|\mathbf{q}\| \|\mathbf{v}\| \cos(\triangle_{\mathbf{q},\mathbf{v}}) \propto \|\mathbf{v}\| \cos(\triangle_{\mathbf{q},\mathbf{v}}) \tag{5.1}$$

where a denotes a latent attribute of items, and $W(q, a)$ represents the weight of query q on attribute a, and $F(v, a)$ represents the score of item v on attribute a. We use $\mathbf{q}$ and $\mathbf{v}$ to denote the vectors of query q and item v, respectively, and $\triangle_{\mathbf{q},\mathbf{v}}$ is the angle between the two vectors. Unfortunately, there is not any existing technique to efficiently solve this problem; a linear search over the set of points appears to be the state of the art.

5.1.1 Parallelization

The online computational cost of recommendation retrieval can be mitigated by parallelization. One possible way of parallelizing involves dividing the queries/users across cores/machines—each worker can compute the recommendations for a single query/user (or a small set of queries/users). Although the parallelization method can reduce the expensive time cost brought by multiple queries, this form of parallelization does not mitigate the high latency of computing recommendations for a single query/user. Our proposed techniques are orthogonal to parallelization, and can be parallelized to improve the scalability. The MT, TA, and AP techniques presented in this chapter aim reducing the single query latency.

5.1.2 Nearest-Neighbor Search

Efficiently finding the top-k recommendation using the dot-product Eq. (5.1) appears to be very similar to the widely studied problem of k-nearest-neighbor search in

[1] http://www.yelp.com.sg/dataset_challenge/.

metric spaces [6]. The k-nearest-neighbor search problem (in metric space) can be solved approximately with the popular Locality-sensitive hashing (LSH) method [3]. LSH has been extended to other forms of similarity functions (as opposed to the distance as a dissimilarity function) like the cosine similarity [1]. In this section, we show that our problem is different from these existing problems.

The problem of k-nearest-neighbor search in metric space is defined as: given a query q, the aim is to find k points $v \in V$ with least Euclidean distance to the query point q, and the Euclidean distance is computed as follows:

$$\| \mathbf{q} - \mathbf{v} \|^2 = \| \mathbf{q} \|^2 + \| \mathbf{v} \|^2 - 2\mathbf{q}^T\mathbf{v} \propto \| \mathbf{v} \|^2 /2 - \mathbf{q}^T\mathbf{v} \neq -\mathbf{q}^T\mathbf{v} \tag{5.2}$$

Obviously, if all the points in V are normalized to the same length, then the problem of top-k recommendation with respect to the dot-product is equivalent to the problem of k-nearest-neighbor search in any metric space. However, without this restriction, the two problems can yield very different answers.

Similarly, the problem of k-nearest-neighbor search w.r.t. cosine similarity is defined as: given a query q, the aim is to find k points $v \in V$ with maximum cosine similarity for the query q, and the cosine similarity is computed as follows:

$$\cos(\triangle_{\mathbf{q},\mathbf{v}}) = \frac{\mathbf{q}^T\mathbf{v}}{\| \mathbf{q} \| \| \mathbf{v} \|} \propto \frac{\mathbf{q}^T\mathbf{v}}{\| \mathbf{v} \|} \neq \mathbf{q}^T\mathbf{v} \tag{5.3}$$

From the above equation, we can see that our problem of top-k recommendation w.r.t. the dot-product is equivalent to the problem of k-nearest-neighbor search w.r.t. cosine similarity, if all the points in the set V are normalized to the same length. Under general conditions, the two problems can be very different.

As analyzed above, our problem is not equivalent to the classic k-nearest-neighbor search problem in metric space or cosine similarity, thus existing solutions (e.g., the LSH family) to the knn problem cannot be straightforwardly applied to our problem. Actually, our problem is much harder than the knn problem. Unlike the distance functions in metric space, dot-products do not induce any form of triangle inequality. Moreover, this lack of any induced triangle inequality causes the similarity function induced by the dot-products to have no admissible family of locality sensitive hashing functions. Any modification to the similarity function to conform to widely used similarity functions (like Euclidean distance or Cosine-similarity) will create inaccurate results.

Moreover, dot-products lack the basic property of coincidence the self similarity is highest. For example, the Euclidean distance of a point to itself is 0; the cosine-similarity of a point to itself is 1. The dot-product of a point q to itself is $\| \mathbf{q} \|^2$. There can possibly be many other points v_i ($i = 1, 2, \ldots$) in the set V such that $\mathbf{q}^T\mathbf{v_i} > \| \mathbf{q} \|^2$. Without any assumption, the problem of top-k recommendation with respect to the dot-product is inherently harder than the previously addressed similar problems.

Algorithm 5: Make-Metric-Tree-Split

Input: Item Set V;

Pick a random item $v \in V$;
$\mathbf{a} \leftarrow arg\ \max_{v' \in V} \parallel \mathbf{v} - \mathbf{v}' \parallel$;
$\mathbf{c} \leftarrow arg\ \max_{v' \in V} \parallel \mathbf{a} - \mathbf{v}' \parallel$;
$\mathbf{w} \leftarrow \mathbf{c} - \mathbf{a}$;
$b \leftarrow \frac{-1}{2}(\parallel \mathbf{c} \parallel^2 - \parallel \mathbf{a} \parallel^2)$;
return $(\mathbf{w}, b)$;

Algorithm 6: Make-Metric-Tree

Input: Item Set V;
Output: Metric Tree T;

$T.V \leftarrow V$;
$T.center \leftarrow mean(V)$;
$T.radius \leftarrow arg\ \max_{v \in V} \parallel T.center - \mathbf{v} \parallel$;
if $|V| \leq N_0$ **then**
  return T; //leaf node
end
else
  //else split the set;
  $(\mathbf{w}, \mathbf{b}) \leftarrow$ Make-Metric-Tree-Split(V);
  $V_l \leftarrow \{v \in V : \mathbf{w}^T \mathbf{v} + b \leq 0\}$;
  $V_r \leftarrow V - V_l$;
  $T.left \leftarrow$ Make-Metric-Tree(V_l);
  $T.right \leftarrow$ Make-Metric-Tree(V_r);
  return T;
end

5.2 Metric Tree

In this section, we describe metric tree and develop a novel branch-and-bound algorithm to provide fast top-k recommendations.

Metric trees [4] are binary space-partitioning trees that are widely used for the task of indexing datasets in Euclidean spaces. The space is partitioned into overlapping hyper-spheres (balls) containing the points. We use a simple metric tree construction heuristic that tries to approximately pick a pair of pivot points farthest apart from each other [4], and splits the data by assigning points to their closest pivot. The tree T is built hierarchically and each node in the tree is defined by the mean of the data in that node ($T.center$) and the radius of the ball around the mean enclosing the points in the node ($T.radius$). The tree has leaves of size at most N_0. The splitting and the recursive tree construction algorithm is presented in Algorithms 5 and 6.

The tree is space efficient since every node only stores the indices of the item vectors instead of the item vectors themselves. Hence, the matrix for the items is

never duplicated. Another implementation optimization is that the vectors in the items' matrix are sorted in place (during the tree construction) such that all the items in the same node are arranged serially in the matrix. This avoids random memory access while accessing all the items in the same leaf node.

5.2.1 Branch-and-Bound Algorithm

Metric trees are used for efficient nearest-neighbor search and are fairly scalable [5]. The search employs a depth-first branch-and-bound algorithm. A nearest-neighbor query is answered by traversing the tree in a depth-first manner—going down the node closer to the query first and bounding the minimum possible distance to items in other branches with the triangle-inequality. If this branch is farther away than the current neighbor candidate, the branch is removed from computation. Since the triangle inequality does not hold for the dot-product, we present a novel analytical upper bound for the maximum possible dot-product of a query vector with points (in this case, items) in a ball. We then employ a similar branch-and-bound algorithm for the purposes of searching for the k-highest dot-products (as opposed to the minimum pairwise distance in k-nearest-neighbor search).

Deriving the Upper Bound. Let $B^r_{v_c}$ be the ball of items centered around v_c with radius r. Suppose that v^* is the best possible recommendation in the ball $B^r_{v_c}$ for the query represented by the vector $\mathbf{q}$, and r^* be the Euclidean distance between the ball center v_c and the best possible recommendation v^* (by definition, $r^* \leq r$). Let $\triangle$ be the angle between the vector $\mathbf{v_c}$ and the vector $\mathbf{v^*} - \mathbf{v_c}$, $\triangle_{q,v_c}$ and $\triangle_{v^*,v_c}$ be the angles between the vector $\mathbf{v_c}$ and vectors $\mathbf{q}$ and $\mathbf{v^*}$, respectively, as shown in Fig. 5.1. The distance of v^* from v_c is $(r^* sin\triangle)$ and the length of the projection of v^* onto v_c is $\parallel \mathbf{v_c} \parallel + r^* cos\triangle$. Therefore, we have:

$$\parallel \mathbf{v^*} \parallel = \sqrt{(\parallel \mathbf{v_c} \parallel + r^* \cos\triangle)^2 + (r^* \sin\triangle)^2} \tag{5.4}$$

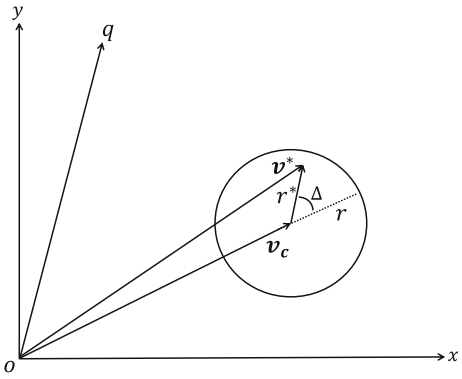

Fig. 5.1 Bounding with a ball

Algorithm 7: Search-Metric-Tree

Input: Item Tree Node T, query q;

```
1  if q.ub < q^T T.center + T.radius · ‖ q ‖ then
2      if isLeaf(T) then
3          for v ∈ T.V do
4              if q^T v > q.ub then
5                  v' ← arg min_{v_i ∈ q.candidates} q^T v_i;
6                  q.candidates ← (q.candidates − {v'}) ⋃ {v};
7                  q.ub ← min_{v_i ∈ q.candidates} q^T v_i;
8              end
9          end
10     end
11     else
12         //best depth first traversal
13         if q^T T.left.center < q^T T.right.center then
14             Search-Metric-Tree(q, T.right);
15             Search-Metric-Tree(q, T.left);
16         end
17         else
18             Search-Metric-Tree(q, T.left);
19             Search-Metric-Tree(q, T.right);
20         end
21     end
22 end
```

$$\cos \triangle_{v^*,v_c} = \frac{\| \mathbf{v_c} \| + r^* \cos \triangle}{\| \mathbf{v^*} \|}, \quad \sin \triangle_{v^*,v_c} = \frac{r^* \sin \triangle}{\| \mathbf{v^*} \|}. \tag{5.5}$$

Let $\triangle_{q,v^*}$ be the angle between the vectors $\mathbf{q}$ and $\mathbf{v^*}$. This gives the following inequality regarding the angle between the query and the best possible recommendation (we assume that the angles lie in the range of $[-\pi, +\pi]$ instead of the usual range $[0, 2\pi]$):

$$|\triangle_{q,v^*}| \geq |\triangle_{q,v_c} - \triangle_{v^*,v_c}|,$$

which implies

$$\cos \triangle_{q,v^*} \leq \cos(\triangle_{q,v_c} - \triangle_{v^*,v_c}) \tag{5.6}$$

since $\cos(\cdot)$ is monotonically decreasing in the range $[0, \pi]$. Using this equality we obtain the following bound for the highest possible affinity between the user and any item within that ball:

$$\max_{v \in B^r_{v_c}} \mathbf{q}^T \mathbf{v} = \mathbf{q}^T \mathbf{v}^* = \| \mathbf{q} \| \| \mathbf{v}^* \| \cos \triangle_{q,v^*} \leq \| \mathbf{q} \| \| \mathbf{v}^* \| \cos(\triangle_{q,v_c} - \triangle_{v^*,v_c})$$

where the last inequality follows from Eq. (5.6). Substituting Eqs. (5.4) and (5.5) in the above inequality, we have:

$$\begin{aligned}
\max_{v \in B^r_{v_c}} \mathbf{q}^T \mathbf{v} &\leq \| \mathbf{q} \| (\cos \triangle_{q,v_c} (\| \mathbf{v_c} \| + r^* \cos \triangle) + \sin \triangle_{q,v_c} (r^* \sin \triangle)) \\
&\leq \| \mathbf{q} \| \max_{\triangle} (\cos \triangle_{q,v_c} (\| \mathbf{v_c} \| + r^* \cos \triangle) + \sin \triangle_{q,v_c} (r^* \sin \triangle)) \\
&= \| \mathbf{q} \| (\cos \triangle_{q,v_c} (\| \mathbf{v_c} \| + r^* \cos \triangle_{q,v_c}) + \sin \triangle_{q,v_c} (r^* \sin \triangle_{q,v_c})) \\
&\leq \| \mathbf{q} \| (\cos \triangle_{q,v_c} (\| \mathbf{v_c} \| + r \cos \triangle_{q,v_c}) + \sin \triangle_{q,v_c} (r \sin \triangle_{q,v_c}))
\end{aligned}$$

The second inequality comes from the definition of maximum, and the next equality comes from maximizing over $\triangle$ giving us the optimal value for $\triangle = \triangle_{q,v_c}$. The last inequality follows the $r^* \leq r$. Simplifying the final inequality gives us the following upper bound:

$$\max_{v \in B^r_{v_c}} \mathbf{q}^T \mathbf{v} \leq \mathbf{q}^T \mathbf{v_c} + r \| \mathbf{q} \| . \tag{5.7}$$

The Retrieval Algorithm. Using this upper bound in Eq. (5.7) for the maximum possible dot-product, we present the depth-first branch-and-bound algorithm to search for the k-highest dot-products in Algorithm 7. In the algorithm, the object $q.candidates$ contains the set of current best k candidate items and $q.ub$ denotes the lowest affinity between the query and its current best candidates. The algorithm begins at the root of the tree of items. At each subsequent step, the algorithm is at a tree node. Using the bound in Eq. (5.7), the algorithm checks if the best possible item in this node is any better than the current best candidates for the query. If the check fails, this branch of the tree is not explored any more. Otherwise, the algorithm recursively traverses the tree, exploring the branch with the better potential candidates in a depth-first manner. If the node is a leaf, the algorithm just finds the best candidates within the leaf with the simple naive search. This algorithm ensures that the exact solution (i.e., the best candidates) is returned by the end of the algorithm.

5.3 TA-Based Algorithm

The straightforward method of generating the top-k items needs to compute the ranking scores for all items according to Eq. (5.1), which is computationally inefficient, especially when the number of items becomes large. To speed up the process of producing recommendations, we extend the Threshold-based Algorithm (TA) [7, 8], which is capable of finding the top-k results by examining the minimum number of items.

Algorithm 8: Threshold-based algorithm

Input: An item set V, a query q and A ranked lists L_a;
Output: List L with all the k highest ranked items;

Initialize priority lists PQ, L and the threshold score S_{Ta};
for $a = 1$ *to* A **do**
  $v = L_a.getfirst()$;
  Compute $S(q, v)$ according to Eq. (5.1);
  $PQ.insert(a, S(q, v))$;
end
Compute S_{Ta} according to Eq. (5.8);
while $true$ **do**
  $nextListToCheck = PQ.getfirst()$;
  $PQ.removefirst()$;
  $v = L_{nextListToCheck}.getfirst()$;
  $L_{nextListToCheck}.removefirst()$;
  if $v \notin L$ **then**
    if $L.size() < k$ **then**
      $L.insert(v, S(q, v))$;
    end
    else
      $v' = L.get(k)$;
      if $S(q, v') > S_{Ta}$ **then**
        break;
      end
      if $S(q, v') < S(q, v)$ **then**
        $L.remove(k)$;
        $L.insert(v, S(q, v))$;
      end
    end
  end
  if $L_{nextListToCheck}.hasMore()$ **then**
    $v = L_{nextListToCheck}.getfirst()$;
    Compute $S(q, v)$ according to Eq. (5.1);
    $PQ.insert(nextListToCheck, S(q, v))$;
    Compute S_{Ta} according to Eq. (5.8);
  end
  else
    break;
  end
end

We first precompute the ordered lists of items, where each list corresponds to a latent attribute learned by the latent-class models. For example, given A latent attributes, we will compute A lists of sorted items (i.e., inverted indices), L_a, $a \in \{1, 2, \ldots, A\}$, where items in each list L_a are sorted according to $F(v, a)$. Given a query q, we run Algorithm 8 to compute the top-k items from the A sorted lists and return them in the priority list L. As shown in Algorithm 8, we maintain a priority list PL of the A sorted lists where the priority of a list L_a is determined by the ranking

score (i.e., $S(q, v)$) of the first item v in L_a (Lines 2–6). In each iteration, we select the most promising item (i.e., the first item) from the list that has the highest priority in PL and add it to the resulting list L (Lines 9–16). When the size of L is no less than k, we will examine the kth item in the resulting list L. If the ranking score of the kth item is higher than the *threshold score* (i.e., S_{Ta}), the algorithm terminates early without checking any subsequent items (Lines 18–21). Otherwise, the kth item v' in L is replaced by the current item v if v's ranking score is higher than that of v' (Lines 22–25). At the end of each iteration, we update the priority of the current list as well as the threshold score (Lines 28–33).

Equation (5.8) illustrates the computation of the threshold score, which is obtained by aggregating the maximum $F(v, a)$ represented by the first item in each list L_a (i.e., $\max_{v \in L_a} F(v, a)$). Consequently, it is the maximum possible ranking score that can be achieved by the remaining unexamined items. Hence, if the ranking score of the kth item in the resulting list L is higher than the threshold score, L can be returned immediately because no remaining item will have a higher ranking score than the kth item.

$$S_{Ta} = \sum_{a=1}^{A} W(q, a) \max_{v \in L_a} F(v, a) \tag{5.8}$$

5.3.1 Discussion

Being different from the metric tree-based algorithm, the TA-based algorithm requires the ranking function defined in Eq. (5.1) to be monotone given a query. Both the ranking functions in the nonnegative matrix factorization models and the probabilistic generative models developed in the previous three chapters meet this requirement, since the query weight on each attribute $W(q, a)$ is nonnegative in these models. It is easy to understand that Algorithm 8 is able to correctly find the top-k items if the ranking function $S(q, v)$ defined in Eq. (5.1) is monotone. Below, we will prove it formally.

Theorem 5.1 *Algorithm 8 is able to correctly find the top-k items if the ranking function $S(q, v)$ defined in Eq. (5.1) is monotone.*

Proof Let L be a ranked list returned by Algorithms 8 which contains the k spatial items that have been seen with the highest ranking scores. We only need to show that every item 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 seen in running Algorithm 8. So assume that v was not seen, and the score of v in each attribute a is $F(v, a)$. For each ranked list L_a, let $\widetilde{v}_a$ be the last item seen in the list L_a. Therefore, $F(v, a) \leq F(\widetilde{v}_a, a)$, for every a. Hence, $S(q, v) \leq S_{Ta}$ where S_{Ta} is the threshold score. The inequality $S(q, v) \leq S_{Ta}$ holds because of the monotonicity of the ranking function $S(q, v)$ defined in Eq. (5.1). But by definition of L, for every v' in L we have $S(q, v') \geq S_{Ta}$. Therefore, for every v' in L we have $S(q, v') \geq S_{Ta} \geq S(q, v)$, as desired.

Besides, Algorithm 8 has another nice property that it is instance optimal with accessing the minimum number of items, and no deterministic algorithm has a lower optimality ratio [2]. We use the word "optimal" to reflect the fact that Algorithm 8 is best deterministic algorithm. Intuitively, instance optimality corresponds to optimality in every instance, as opposed to just worst case or the average case. There are many algorithms that are optimal in a worst-case sense, but are not instance optimal.

Below, we will investigate the instance optimality of Algorithm 8 by an intuitive argument. If P is an algorithm that stops earlier than Algorithm 8 in a certain case, before P finds k items whose ranking score is at least equal to the threshold score S_{Ta}, then P must make a mistake, since the next unseen item v might have a ranking score equal to $F(\widetilde{v}_a, a)$ in each attribute a, and hence have ranking score $S(q, v) = S_{Ta}$. This new item, which P has not even seen, has a higher ranking score than some item in the top-k list that was output by P, and so P erred by stopping too soon.

5.4 Attribute-Pruning Algorithm

Both metric-tree and TA algorithms focus on pruning item search space, and they cannot reduce the time cost of computing the ranking score for a single item. Moreover, they cannot adapt to the high-dimension data, i.e., the number of latent attributes is large. When the dimensionality of items is high (e.g., $A > 500$), the tree index structures and tree-based search algorithms (e.g., metric-tree and R-tree) will lose their pruning ability, as analyzed in [5], which is also validated in our experiments. As for TA, it 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 down the efficiency of TA when the dimensionality is high.

To overcome the curse of dimensionality and speed up the online recommendation, we propose an efficient algorithm to prune the attribute space and facilitate fast computation of the ranking score for a single POI, inspired by TA algorithm [8] and Region Pruning strategy [9]. Our algorithm is based on three 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; (2) items with high values on these preferred attributes are more likely to become the top-k recommendation results; and (3) the attribute values of most items also exhibit sparsity, i.e., each POI has significant values for only a handful of attributes.

The above three observations indicate that only when a query prefers an attribute and the item has a high value on that attribute, will the score $W(q, a)F(v, a)$ contribute significantly to the final ranking score. Thus, we first pre-compute ordered lists of items, where each list corresponds to a latent attribute. For example, given A latent attributes, we will compute A lists of sorted items, L_{a_j}, $1 \leq j \leq A$, where items in each list L_{a_j} are sorted according to $F(v, a_j)$. Different from the threshold algorithm (TA) developed in [8], each sorted list L_{a_j} only stores k items with highest $F(v, a_j)$ values instead of all items. Hence, it is space-saving. Besides, for each item v, its attributes are preranked offline according to the value of $F(v, a_j)$. Given an

Algorithm 9: Attribute Pruning Algorithm

Input: An item set V, a query q and A ranked lists L_a;
Output: Result list L with k highest ranking scores;

Initialize L as $\emptyset$;
Sort the query attributes by $W(q, a)$;
Choose top m attributes satisfying: $\sum_{j=1}^{m} W(q, a_j) > 0.9 \sum_{j=1}^{A} W(q, a_j)$;
for $j = 1$ *to* m **do**
  for $v \in L_{a_j}$ *and* $v \notin L$ **do**
    Compute $S(q,\ v)$ according to Eq. (5.1);
    if $L.size() < k$ **then**
      $L.add(< v, S(q,\ v) >)$;
    end
    else
      $v' = L.top()$;
      if $S(q,\ v) > S(q,\ v')$ **then**
        $L.removeTop()$;
        $L.add(v, S(q,\ v))$;
      end
    end
  end
end
for $v \in V$ *and* $v \notin L$ **do**
  $PS = 0$, $PW = 0$, $Skip = false$, and $v' = L.top()$;
  while *there exists attribute a not examined for v* **do**
    $a = v.nextAttribute()$;
    $PS = PS + W(q, a)F(v, a)$;
    $PW = PW + W(q, a)$;
    if $PS + (\sum_{j=1}^{A} W(q, a_j) - PW)F(v, a) \leq S(q, v')$ **then**
      $Skip = true$;
      $break$;
    end
  end
  if $Skip == false$ **then**
    if $S(q, v) > S(q, v')$ **then**
      $L.removeTop()$;
      $L.add(< v, S(q, v) >)$;
    end
  end
end
$L.Reverse()$;
Return L;

online query q, we develop a branch and bound algorithm, as shown in Algorithm 9, to prune the search space of the attributes in the computation of the ranking score, i.e., after we have scanned a small number of significant attributes for an item, it may not be necessary to examine the remaining attributes. The algorithm is called AP (Attribute Pruning) and contains two components: *initialization* and *pruning*.

In the initialization component (Lines 1–18), we select k candidate items that are potentially good for recommendation. Specifically, we pick top m attributes which cover most the query's preferences with smallest m, i.e., $\sum_{j=1}^{m} W(q, a_j) > \rho \sum_{j=1}^{A} W(q, a_j)$, where ρ is a predefined constant between 0 and 1 (Line 3). In our experiment, AP achieves its best performance for $\rho = 0.9$. For each of the top m attributes a, we choose top ranked items from L_a as candidates (Lines 4–18).

In the pruning component (Lines 19–36), we check whether we can avoid traversing unnecessary attributes for item v according to the descending order of $F(v, a)$. Suppose we have traversed attributes $\{a_1, \ldots, a_{i-1}\}$. The partial score we have computed for the traversed attributes is

$$PS(q, v) = \sum_{j=1}^{i-1} W(q, a_j) F(v, a_j).$$

When we explore the ith attribute, we compute the upper bound of ranking score for the item v as:

$$UB(q, v) = PS(q, v) + \sum_{j=i}^{A} W(q, a_j) F(v, a_i) \tag{5.9}$$

Because we check the attributes in the descending order of $F(v, a)$, the actual value of $F(v, a)$ for the remaining attributes should be less than the value for the current attribute, i.e., $F(v, a_i)$. Therefore, we have a partial ranking score for the rest of the attributes, which is at most

$$\sum_{j=i}^{A} W(q, a_j) F(v, a_i), \tag{5.10}$$

where $\sum_{j=i}^{A} W(q, a_j)$ is the portion of query preferences for the rest attributes. The upper bound of $\sum_a W(q, a)\ F(v, a)$ for all attributes is $PS(q, v) + \sum_{j=i}^{A} W(q, a_j)$ $F(v, a_i)$, which results in Eq. (5.9).

We employ a binary min-heap to implement L so that the top item v' has the smallest ranking score in L (Line 20). If the upper bound is smaller than the ranking score of v', we skip the current item (no need to check the remaining attributes) (Lines 25–28). Otherwise, we continue to check the remaining attributes. If all attributes are examined for the item and the item is not pruned by the aforementioned upper bound, we obtain the full score of the item to compare with v' (Lines 30–35). We remove the item v' and add the current item to the list if its full score is larger than v' (Lines 31–34).

Note that our proposed AP algorithm in this section also requires the ranking function $S(q, v)$ to be monotone, e.g., the query weight on each attribute $W(q, a)$ is nonnegative.

5.5 Experiments

In this section, we evaluate the top-k recommendation efficiency of metric tree (MT), TA, and attribute pruning (AP) algorithms on both real-world and large-scale synthetic datasets. There are 114,508 users, 62,462 spatial items and 1,434,668 check-ins in the real-world dataset Twitter. Every user has approximately 13 check-ins and every spatial item is associated with 23 check-ins on average. The users' check-ins display the power-law distribution. To keep the sparsity property on the synthetic dataset, we generate a dataset with 23 million check-ins, 1 million spatial items and 1.77 million users to simulate the distribution of the check-ins on the real-world dataset.

5.5.1 Experimental Results

This experiment is to evaluate the efficiency of our proposed online recommendation algorithms MT, TA, and AP on both the real-life and large-scale synthetic datasets. We compare them with a baseline algorithms. The baseline is a brute-force algorithm (BF) that needs to compute a ranking score for each item and selects top-k ones with highest ranking scores. All the online recommendation algorithms were implemented in Java (JDK 1.7) and run on a Windows Server 2008 with 256G RAM.

Figure 5.2 shows the time costs of producing top-10 recommendation on the large-scale synthetic dataset. We control the number of available items to vary from 0.1 million to 1 million to test the scalability, and the dimensionality is set to be 10, 50, 100, 500, 1000, and 1500. From the results, we can observe that these four algorithms have significant performance disparity when the dimensionality increases from 10 to 1500. Obviously, AP exhibits highly desirable scaling characteristics—sub-linear time complexity to both data size and data dimensionality, while other competitor methods, MT and TA, are very sensitive to the data dimensionality, and they perform better than BF only for low-dimensionality setting (e.g., less than 100 dimensionality). The test results also provide important insights to choose online recommendation algorithms: when the data dimensionality is not larger than 10, MT is the best choice; when $10 < A \leq 50$, TA can achieve best performance; and when $A > 50$, we suggest to choose AP algorithm.

To further analyze these algorithms in the high-dimension setting, we test them on the Twitter dataset. Table 5.1 shows the performance with 1500 latent dimensions (i.e., $A = 1500$) and k (i.e., the number of recommendations) set to 1, 5, 10, 15, and 20. A greater value of k is not necessary for the top-k recommendation task. Obviously, the AP algorithm outperforms others significantly and consistently for different number of recommendations. For example, on average the AP algorithm finds the top-10 recommendations from about 62,000 items in 67.57 ms, and achieves 1.65 times faster than the brute-force algorithm (BF).

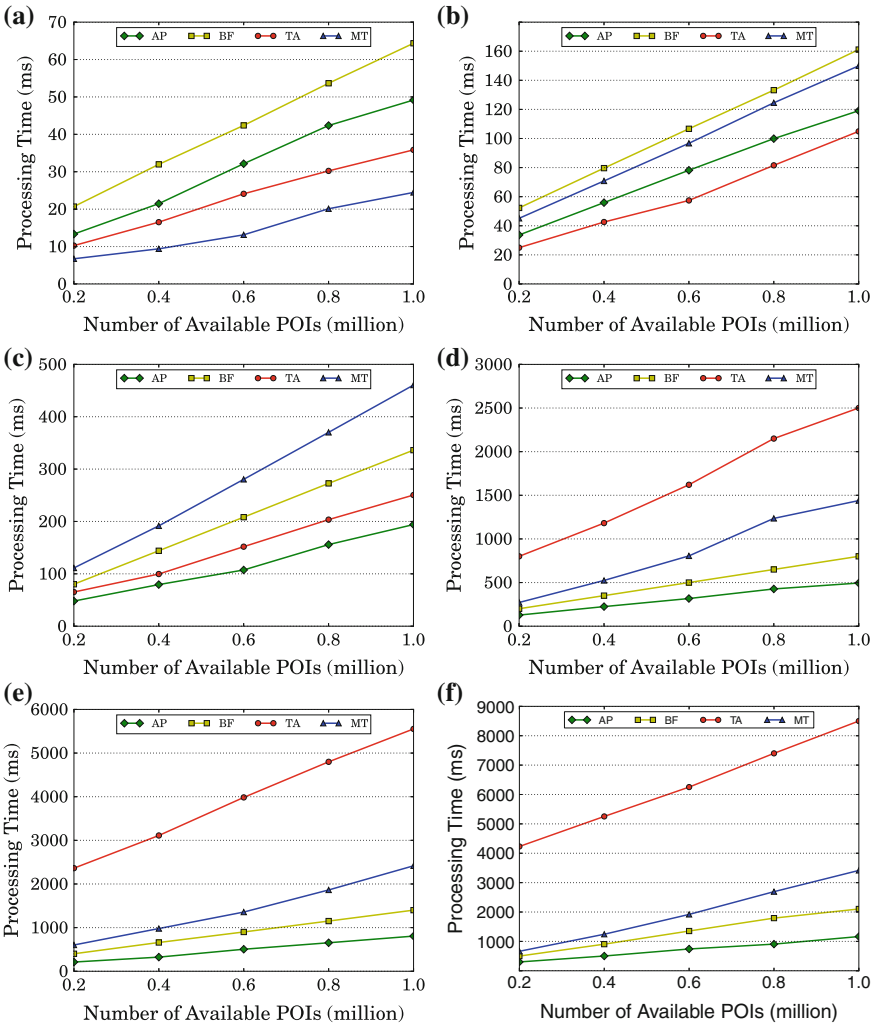

Fig. 5.2 Recommendation Efficiency on the Synthetic Dataset with Varying Dimensionality (A). **a** A = 10. **b** A = 50. **c** A = 100. **d** A = 500. **e** A = 1000. **f** A = 1500

Specifically, from the results, we observe that: (1) AP outperforms BF significantly, justifying the benefits brought by pruning attribute space. It only needs to access very few attributes for each item to compute its partial score, about 145 attributes on average (that is less than 10 %) for $k = 10$, and 120 attributes for $k = 5$, since AP algorithm takes full advantage of the sparsity of both query and POI vectors. (2) Although the time cost of AP 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 naive linear

Table 5.1 Recommendation efficiency on twitter dataset

Methods	Online recommendation time cost (ms)				
	k=1	k=5	k=10	k=15	k=20
AP	**50.91**	**61.75**	**67.57**	**72.13**	**75.32**
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

scan method (BF) in this task, although it achieves better performance than BF in the low-dimension setting. This is because MT loses its ability to prune item search space and needs to scan all items in the leaf nodes when the dimensionality is high. (4) The threshold algorithm (TA) performs worse than the brute-force algorithm, since it still needs to access many items (around 40 % of the items on average for $k = 10$ and 35 % for $k = 5$). Moreover, 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 down the efficiency of TA when the dimensionality is high. In summary, although both TA and MT can achieve better performance than BF due to their ability of pruning POI search space when the dimensionality is not very high, they cannot overcome the curse of dimensionality when the items have thousands of attributes. In contrast, the AP algorithm is designed for pruning attribute space, thus it can still achieve superior performance for the setting of high dimensionality.

5.6 Summary

In this chapter, we proposed three techniques for efficient spatiotemporal recommendation: (i) metric tree (MT). We index the item vectors in a binary spatial partitioning metric tree and used a simple branch-and-bound algorithm with a novel bounding scheme to prune the item search space. (ii) TA. We precomputes an inverted list for each latent attribute a in which items are sorted according to their values on attribute a, and also maintains a priority queue of the inverted lists that controls which inverted list to access in the next. The algorithm has the nice property of terminating early without scanning all items. (iii) Attribute Pruning (AP). Being different from metric tree and TA which focus on pruning item search space, the AP aims to reduce the time cost of computing the ranking score for a single item by pruning the attribute space. We evaluated our algorithms on both real-world and large-scale synthetic datasets, demonstrating the superiority of MT, TA, and AP over the linear-scan algorithm. Moreover, we found that these three techniques show different performances with varying the data dimensionality (i.e., the number of items' latent attributes): when the data dimensionality is not larger than 10, MT is the best choice; when the number of latent attributes is between 10 and 50, TA can achieve best performance; and when the data dimensionality exceeds 50, we suggest to choose the AP.

References

1. Charikar, M.S.: Similarity estimation techniques from rounding algorithms. In: STOC, pp. 380–388 (2002)
2. Fagin, R., Lotem, A., Naor, M.: Optimal aggregation algorithms for middleware. In: PODS, pp. 102–113 (2001)
3. Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In: STOC, pp. 604–613 (1998)
4. Koenigstein, N., Ram, P., Shavitt, Y.: Efficient retrieval of recommendations in a matrix factorization framework. In: CIKM, pp. 535–544 (2012)
5. Liu, T., Moore, A.W., Yang, K., Gray, A.G.: An investigation of practical approximate nearest neighbor algorithms. In: NIPS, pp. 825–832 (2004)
6. Shakhnarovich, G., Darrell, T., Indyk, P.L: Nearest-Neighbor Searching and Metric Space Dimensions, pp. 15–59. MIT Press, Cambridge (2006)
7. Yin, H., Cui, B., Chen, L., Hu, Z., Zhou, X.: Dynamic user modeling in social media systems. ACM Trans. Inf. Syst. **33**(3):10:1–10:44 (2015)
8. Yin, H., Cui, B., Sun, Y., Hu, Z., Chen, L.: Lcars: a spatial item recommender system. ACM Trans. Inf. Syst. **32**(3):11:1–11:37 (2014)
9. Zhao, K., Cong, G., Yuan, Q., Zhu, K.: Sar: a sentiment-aspect-region model for user preference analysis in geo-tagged reviews. In: ICDE (2015)