Effective and efficient topic mining and exploration from geo-textual data

Zhao, Kaiqi

2018

Zhao, K. (2018). Effective and efficient topic mining and exploration from geo-textual data.
Doctoral thesis, Nanyang Technological University, Singapore.

<http://hdl.handle.net/10356/73632>

<https://doi.org/10.32657/10356/73632>

Effective and Efficient Topic Mining and Exploration from Geo-Textual Data

A thesis submitted
for a degree of Doctor of Philosophy
by

ZHAO Kaiqi

School of Computer Science and Engineering
Nanyang Technological University

Aug 2017

Abstract

With the prevalence of online social media (e.g., Facebook, Twitter), location-based services (e.g., Foursquare, Yelp, Flickr), and GPS-enabled devices, a huge number of documents with spatial information are being generated. Such documents are associated with either points of interest (e.g., restaurants) or latitude-longitude coordinates. We call these documents *geo-textual documents*. Geo-textual documents often contain information that indicates public/individual views and interests. It is of great interest to mine and explore topics from geo-textual documents to help various practical tasks, e.g., business analytics, point-of-interest (POI) recommendation, user recommendation, topic exploration, etc.

There are two types of studies on mining topics from geo-textual data — (1) discovering topics of individuals from POI-associated posts (e.g., check-ins); and (2) mining and exploring topics of regions from geo-tagged microblogs. However, both types of studies have several limitations. Firstly, the topics of individuals that are mined from geo-textual data are successfully applied to POI recommendation, location prediction, etc. However, most of the existing methods mine topics from check-in datasets from Foursquare. Because each check-in often consists of limited textual information, and most of the users only shared few check-ins, it is difficult to discover meaningful topics of individuals from the check-in data. Moreover, the existing methods cannot capture topical aspects, e.g., the “environment” of a restaurant, thus failing to tell users why a POI is recommended to the user. Worse still, the existing methods are frequency-based (the more a topic is mentioned, the more likely a user prefers the topic), while ignoring the user’s sentiment. A user may hold negative opinions on some topics even though he/she mentions them many times. Secondly, the existing studies on learning topics of regions only allow users to explore the topics in predefined regions and time spans. A user may want to query topics within a specified region and time span. For example, a social scientist may want to find out breaking events by submitting regions and time spans in an exploratory manner. Some studies propose to learn geographical topic models to uncover latent regions and geographical topics. However, training these models is time consuming. It often takes months to train a model of moderate size (e.g., thousands of topics and thousands of regions) on millions of documents. However, there exists no distributed solution for training geographical topic models.

To overcome the limitations in mining topics of individuals, we address two research challenges. First, we propose an approach to associating POIs with geo-tagged microblogs

to compose a complementary “check-in” data source for topic mining of individuals. Second, we propose a unified model for learning topical aspects and regions of individuals with consideration of sentiment. The proposed model is able to improve the effectiveness of many downstream applications, e.g., POI recommendation, user recommendation, aspect satisfaction analysis, etc.

To overcome the limitations in mining topics of regions, we consider two research problems. First, we develop a framework for exploring topics within a user specified region and time span. The framework can return topics fall in the spatio-temporal query to a user within seconds. Second, to allow efficient training of geographical topic models, we propose a distributed solution that supports learning large geographical topic models with millions of parameters from tens of gigabytes of geo-textual documents within 20 hours on a small cluster of 20 machines.

Acknowledgments

I would like to express my sincere gratitude to my supervisor Prof. Gao Cong, for his patient guidance and elaborative supervision. I learn from him not only the cutting-edge techniques of data mining and database, but also how to do research and how to give good presentations on scientific topics. He also kindly shares his experience on his professional career. Prof. Cong is not only my supervisor but also a good friend in my life. I also would like to thank Prof. Aixin Sun and Dr. Gang Wang for their kind suggestions to my research.

I am truly appreciate my colleagues and friends. They share invaluable ideas and provide great help to my research. They include Xin Cao, Quan Yuan, Lisi Chen, Shanshan Feng, Nguyen Tuan Anh Pham, Kaiyu Feng, Tao Guo, Zhida Chen, Xutao Li, Zhen Hai, Youer Pu, Miao Li, Miao Xie, Zongcheng Ji, Guangtao Wang, Daniel Rugeles Barbosa, Seyed Ali Majid Zonoozi, Xiucheng Li, Yiding Liu, Jin Yao Chin, Tran Dang Quang Vinh, and Yue Chen.

Last but not least, I would like to thank my parents and my girl friend for their support and love all the time.

Contents

Abstract	i
Acknowledgments	iii
List of Figures	viii
List of Tables	xi
1 Introduction	1
1.1 Background and Motivations	1
1.2 Problems and Research Scope	4
1.3 Approach and Methodology	8
1.4 Summary of Contributions	12
2 Literature Survey	14
2.1 Topic Models & Geographical Topic Models	14
2.1.1 Topic Models	14
2.1.2 Geographical Topic Models	16
2.1.3 Distributed Training of Topic Models	20
2.2 Online Data Analytics & Exploration	21
2.2.1 Data Cube & Topic Cube	21
2.2.2 Content Exploration	22
2.3 Location Identification of Documents	22
2.4 User Mobility & Interest Modeling	23
2.4.1 User Behaviour Models	23
2.4.2 Sentiment Analysis	24

3	Mining and Exploring Topics in Regions	26
3.1	Problem Definition	26
3.1.1	Data Model	27
3.1.2	Problem Definition	27
3.2	Framework Overview	29
3.3	Algorithms	30
3.3.1	Combining Topics of Two Octree Cells	30
3.3.2	Processing Partially Covered Cells	35
3.3.3	Octree based Pre-Computation	36
3.3.4	Topic Mining within Region and Time Interval	39
3.4	Experiments	39
3.4.1	Experimental Setup	40
3.4.2	Pre-computation Cost	45
3.4.3	Evaluation on Different Parameters	46
3.4.4	Evaluation on Different Regions and Timespans	49
3.4.5	Qualitative Evaluation on Extracted Topics	49
3.5	Summary	50
4	Distributed Training of Geographical Topic Models	52
4.1	Problem Definition	53
4.1.1	Data and Model Formulation	53
4.1.2	Distributed Training of Geographical Topic Model	53
4.2	Distributed Training for Geographical Topic Models	53
4.2.1	Baseline Solution and Overview of the Proposed Solution	54
4.2.2	The Proposed Training Algorithm & Model Parallelism	57
4.2.3	Parameter Localization	61
4.2.4	Reducing Training Complexity	64
4.3	Implementation	67
4.4	Experiments	68
4.4.1	Experimental Setup	68
4.4.2	Scalability	70

4.4.3	Computation v.s. Network Communication	73
4.4.4	Memory Requirement on Worker Machines	74
4.4.5	Network Communication Cost	74
4.4.6	Sampling Efficiency and Effectiveness in Single Machine	75
4.4.7	Case Study	78
4.5	Summary	79
5	Annotating POIs with Geo-tagged Microblogs	81
5.1	Problem Definition	81
5.2	Basic Solutions	82
5.3	Supervised Bayesian Model	84
5.3.1	Motivation & Novelty	84
5.3.2	Generative Story for Tweets	85
5.3.3	Parameter Learning	88
5.3.4	Prediction for New Tweets	90
5.3.5	Incorporating Tips	91
5.4	Experiments	92
5.4.1	Experimental Setup	92
5.4.2	Parameter Selection	94
5.4.3	Overall Performance	96
5.4.4	Performance of POI-Relatedness Problem	98
5.4.5	Empirical Study on Latent Regions	99
5.5	Summary	101
6	Mining Topical Aspects and Regions of Individuals	102
6.1	Problem Definition	103
6.2	Sentiment-Aspect-Region Model	103
6.2.1	Intuitions	103
6.2.2	Learning Algorithm	106
6.2.3	Efficiency Analysis	109
6.3	Applications	110
6.3.1	POI recommendation	110

6.3.2	User Recommendation	113
6.3.3	Aspect Satisfaction in Regions	114
6.4	Experiments	114
6.4.1	Experimental Setup	114
6.4.2	Model Selection	115
6.4.3	POI recommendation	116
6.4.4	Explanation of Recommendation	120
6.4.5	User Recommendation	121
6.4.6	Aspect Satisfaction in Regions	122
6.4.7	Discussion on Topical Regions	123
6.5	Summary	125
7	Conclusions and Future Work	127
7.1	Conclusions	127
7.2	Future Directions	129
A	List of Publications	131
	References	133

List of Figures

1.1	Examples of topical regions	3
1.2	An example of Foursquare check-in	4
1.3	Topics within a bounded region and a time interval	5
2.1	An example of geographical topic model	17
2.2	Graphical representation of a typical geographical topic model	18
3.1	An example of TMRT query	27
3.2	Framework for TMRT query	31
3.3	Offline pre-computation cost w.r.t. accuracy threshold τ_e	42
3.4	Offline pre-computation cost w.r.t. space threshold τ_s	43
3.5	Offline pre-computation cost w.r.t. number of topics K	43
3.6	Runtime w.r.t. accuracy threshold τ_e	44
3.7	Perplexity w.r.t. accuracy threshold τ_e	44
3.8	Runtime w.r.t. space threshold τ_s	45
3.9	Perplexity w.r.t. space threshold τ_s	45
3.10	Runtime w.r.t. number of topics K	46
3.11	Perplexity w.r.t. number of topics K	46
3.12	Runtime w.r.t. region size	47
3.13	Perplexity w.r.t. region size	47
3.14	Runtime w.r.t. timespan size	48
3.15	Perplexity w.r.t. timespan size	48
3.16	Topics extracted by different methods	50

4.1	Training effectiveness and efficiency (in 100 iterations) of different proposals on a random sample of 10K geo-tagged documents from Twitter dataset. The number of topics and regions are set at 10. Uniform is the uniform proposal, and PGeoTopic is our method. The numbers N in PGeoTopic-N and Uniform-N are metropolis steps.	56
4.2	An example of model parallelism.	61
4.3	An example of transforming a distribution to an alias table	65
4.4	Pipeline to hide I/O and network time cost.	68
4.5	Likelihood v.s. training time on 20 machines.	70
4.6	Varying number of topics	71
4.7	Varying number of regions	71
4.8	Varying number of machines	73
4.9	Varying data size	73
4.10	Training time breakdown - average time cost of network communication v.s. computation per iteration.	74
4.11	Average network communication time (in sec.) per iteration.	75
4.12	Average size of the topic-region parameters (in GB) transmitted via network per iteration.	75
4.13	Log-likelihood growth along iterations	76
4.14	Log-likelihood growth along time	77
4.15	Training time of different number of topics ($R = 10$)	77
4.16	Training time of different number of regions ($K = 10$)	78
4.17	Four example regions learned on the Twitter dataset. The blue ovals are the contour lines of the Gaussian distributions with confidence level $p = 0.9$. The red pins are the centers of the regions.	79
5.1	Supervised Bayesian Model for POI Annotation	86
5.2	Effects of the number of dummy POIs (fixing the number of latent regions at 50)	95
5.3	Effects of the number of Latent Regions (fixing the number of dummy POIs to be 10,000)	96

5.4	Precision, Recall, and F1 of all the 9 methods, on the two datasets NYC and SG	97
5.5	R-Precision, R-Recall, and R-F1 of the four methods for the POI-Relatedness problem, on the two datasets	99
5.6	Regions in both Datasets (sBM v.s. k-means)	100
6.1	Sentiment-Aspect-Region model (SAR)	105
6.2	Precision@N and Recall@N for All-category POI recommendation	118
6.3	MAP for All-category POI recommendation	119
6.4	Precision@N and Recall@N for Single-category POI recommendation . .	120
6.5	MAP for Single-category POI recommendation	121
6.6	All-category POI recommendation time consumption	122
6.7	The region preference of user 64 and the recommended POIs	124
6.8	Precision@N and Recall@N for user recommendation	125
6.9	MAP for user recommendation	126

List of Tables

2.1	Word tokens of documents. The topic estimated for each word token is placed in the brackets after the word token. k_1 and k_2 stand for the movie topic and food topic, respectively.	17
3.1	Information of documents	28
3.2	Final topic assignment of R_1	28
3.3	Final topic assignment R_2	29
3.4	Dataset statistics	42
3.5	Parameter settings	44
4.1	Summary of notations	54
4.2	Statistics of the datasets	69
4.3	Memory usage of PGeoTopic in gigabytes	74
4.4	Topics for each region	80
5.1	Summary of notations	87
5.2	Statistics of the two datasets	93
5.3	Summary of the 9 methods evaluated in our experiments. The features used by these methods are listed (U – User preference, R – Latent regions)	94
6.1	Description of symbols	104
6.2	Statistics of the two datasets	115
6.3	User Aspect Preference & Positive Aspects of Recommended POIs	123
6.4	Aspect satisfaction of “Service” of “Beauty & Spas” in two regions	123
6.5	Top 15 representative words for three regions	124
6.6	Representative words for aspect and sentiment	126

Chapter 1

Introduction

1.1 Background and Motivations

Social media (e.g., Facebook, Twitter) have gained rapid development in the past decade. In the early stage, these platforms attracted users to share their status, comments on news events and express their opinion on different topics. As the proliferation of GPS devices, notably smart phones, these social media platforms start to enable users to associate a pair of latitude-longitude coordinates for each post shared. In addition, location-based services (e.g., Foursquare, Yelp, Flickr) allow users to post their location in the form of “check-ins” to points of interest (e.g., restaurants), and share their opinions in the form of tips or reviews to the location. We refer to the geo-tagged social media posts, check-ins, geo-tagged tips/reviews as *geo-textual documents*. Geo-textual documents are being generated in an unprecedented scale. For example, Twitter has 320 million monthly active users who composed 500 million tweets daily¹, 3.1% of which contain the user’s location information [SM15]. For another example, Foursquare has surpassed a total of 10 billion check-ins and has roughly 9 million check-ins generated per day².

As the geo-textual documents connect the virtual world of social media to real world places (e.g., restaurants, parks), geo-textual documents often cover a wide range of individual/public views and interests with geographical distributions. For example, a Twitter user in Boston may be interested in entirely different matters and news compared to a user in Hong Kong. Also, the trending topics in Boston will be different from those in U.S.A., as the trending topics in Boston may have more regional characteristics (e.g., Boston Marathon). The characteristics of geo-textual documents open a great opportunity to understand user’s behaviors, opinions for improving services and products. In the past several years, the online social media and location-based services provide new features by mining geo-textual documents shared by the users to retain active users and

¹<http://www.internetlivestats.com/twitter-statistics/>

²<https://foursquare.com/about>

merchants. For example, Foursquare makes use of the user’s submitted “check-in” data to provide location recommendation for users in order to help users better select a place (e.g., restaurant) to visit. Apart from aforementioned features, market analysis, customer relationship management, sale promotions, etc., are also important applications for users and merchants.

A smart location-based service should consider the needs from both users and merchants. For example, users may want to find out interesting places from a large collections of places to visit, and post comments on the places. Merchants may want to analyze what are the hot topics within a region in order to do promotion, market analysis, etc. Therefore, a desired location-based service platform should be able to understand individual users and spatial regions using the geo-textual data, such that a sound business cycle can be established – Improving the user experience brings benefits to merchants and merchants can develop and improve their business based on the feedback of user experience to better serve the users.

To understand spatial regions, we often need to uncover the topics reside in the regions. The regions can either be provided by users as queries or be automatically learned from the data. In the first scenario, a user may want to get a summary from the documents in a region, because there can be millions of documents in a region (e.g., tweets around New York city) and the data is too “big” for a user to understand the region. Such data summarization helps many applications. For example, business owner can discover topics that are mostly talked about on their products at different locations in order for making strategy for improvement and advertisement, and news providers can find breaking events from tweets within a region. Existing work on Online Analytical Processing(OLAP) mines topics from geo-textual data by building a data warehouse like topic cube [ZZH09] to allow exploring topics in cubes with predefined region (e.g., city) and time span. However, in many applications, e.g., market analysis, a user often wants query a spatial area that crosses multiple predefined regions in topic cube. Topic cube is not flexible enough for finding topics within such arbitrary regions and time spans. There exist no solution for supporting topic exploration in a user specified query region and timespan.

In the second scenario, topics and regions are learned simultaneously in order to support topic comparison between different topical/functional regions [Siz10, YCH⁺11, AHS13, LEHC15] as illustrated in Figure 1.1. Different from topic cube in which regions are discrete and predefined (e.g., city), topical regions are often continuous and automatically detected by the geographical topic models. Different topical regions (represented by ovals) may be featured by different geographical topics, each of which is a word distribution. A region like “beach” is often featured by words “dressing”, “drinking”, etc., while the “university” region may be probably related to words “campus”, “class”, etc. However, training geographical topic models is extremely time consuming,

since the training complexity is proportional to both data size and model size. As observed in our experiments, training a moderate geographical topic model with only 10 topics and 60 regions in a dataset with 120 million word tokens took more than 52 hours. In practice, both the number of topics and the number of regions can be hundreds of thousands in a wide-range dataset (e.g., US), resulting in a model with more than 40 billion parameters. Worse still, it may take months to train a model of moderate size (e.g., thousands of topics on thousands of regions) on millions of geo-textual documents. Unfortunately, there exist no distributed solution for training large geographical topic model on big geo-textual document collections.

Figure 1.1: Examples of topical regions

To understand individual users from geo-textual data, we need to uncover the user interests on different topics and regions that are learned based on the aforementioned geographical topic modeling techniques. Mining the topical and regional interests of users supports many applications, e.g., POI recommendation, POI annotation, location prediction, etc. Most of the existing work [YCM⁺13, YZS⁺15, WYC⁺15, YCZ⁺15, ZHSS16] mine the regions, topics and user interests from check-in datasets such as Foursquare and Gowalla. Because each check-in only consists of limited textual information (Figure 1.2), and most of the users shared few check-ins, it is difficult to mine meaningful topics of individuals only from the check-in data. Observe from our collected Twitter datasets, more than 20% geo-tagged tweets are related to POIs. We argue that associating the large geo-tagged text resources such as geo-tagged microblogs to POIs can largely enrich the check-in records of POIs, especially when these geo-tagged microblogs grow at an unprecedented speed. However, no existing work considers such “data integration”. Moreover, we observe that geo-textual data often contains abundant information of user opinions on different aspects of locations. Uncovering such fine-grained topics (e.g., aspect) of individuals has greatly improved recommendations in the e-commerce domain.

These fine-grained topics are also helpful in explaining why a POI is recommended (e.g., because of service or environment) to a particular user in POI recommendations. However, the existing methods mainly focus on mining general topics of individual users (e.g., entertainment, work), and thus they are not able to capture user interests on topical aspects (e.g., service, environment). In addition, when modeling the topic interests of individual users, existing methods are based on historical check-ins (the more a topic is mentioned, the more likely a user prefers the topic), while ignoring the user’s sentiment. A user may hold negative opinions on some topics even though they mention the topic many times.

Figure 1.2: An example of Foursquare check-in

My research aims at addressing the aforementioned limitations and introducing several novel technologies of mining topics of both individual and regions. By efficiently and effectively learning the topics of regions, the location-based provider can suggest the business owners quick solution for improving their business. Based on the topics and regions, by understanding the topical and regional interests of individual users effectively, it is easier for a location-based provider to develop personalized applications (e.g., recommendations).

1.2 Problems and Research Scope

To address the limitations in mining topics of regions, we first study the problem of mining and exploring topics in regions in which the region are submitted as queries. Then, we consider efficient distributed training of geographical topic models where we learn the topics and regions automatically. To address the limitations in mining topics of individuals, we first annotate POIs with geo-tagged microblogs to enrich the textual information of POIs. Then, we study the problem of mining user interests on topical aspects in addition to regions with consideration of sentiment.

Mining and exploring topics in regions

Topic exploration provides users insights about events, trends, and public concerns within the specified region and time span. However, there lacks solution for efficiently answering such spatio-temporal queries on topics. We propose a type of query for Topic Mining over documents within a specified Region and a Timespan (TMRT) which finds K topics from the documents fall in the user specified rectangular region and time span. Figure 1.3 exemplifies the problem of topic exploration, where a user draws a bounded region (which is represented by the purple box on the map) and a time interval, July 4th, 2015. The blue icons indicate the spatio-temporal documents (e.g., tweets or news) containing text, location, and time information. The three topics extracted from the collection of spatial-temporal documents falling in the region on July 4th, 2015 are presented on the right. Since each topic is characterized by a probabilistic distribution over words, we use “Word Clouds” to visualize each topic. Word clouds use larger font to represent higher probability of a word in the topic.

Figure 1.3: Topics within a bounded region and a time interval

It is challenging to mine topics within a given region and a timespan efficiently and effectively. A straightforward method works as follows: given a user specified region and timespan, we train a topic model on the documents falling in the region and the timespan using the existing techniques, such as LDA [BNJ03]. However, mining topics from spatio-temporal documents is very time-consuming. The original problem of training topic models is intractable and collapsed Gibbs sampling is often used to approximately estimate the probabilistic distribution over words for each topic. However, the complexity of collapsed Gibbs sampling is proportional to both data size and model size (i.e., number of topics). For example, learning 100 topics from our three-month Twitter dataset in the New York city region, which contains 0.25 million tweets, took 8.31 minutes. As our Twitter data is a 1% sample of the geo-tagged tweet data, if we extend the result to a collection of all the three-month geo-tagged tweets the New York

city region based on the linear complexity, it would take 13.85 hours to learn 100 topics from the New York city region. Several studies on parallel implementation of LDA [WBS⁺09, SN10, AAG⁺12, LARS14, WZS⁺14, YGH⁺15] try to accelerate the training process. These solutions often requires high cost computation resources but achieves at most sub-linear speedup due to the heavy network communication for parameter synchronization. In this problem, we propose a framework to learn topics online by making use of offline pre-trained models to support efficient topic exploration.

Distributed training of geographical topic models

Existing work on geographical topic models [Siz10, YCH⁺11, LFYX13, YCM⁺13, AHS13, LXP⁺15, LEHC15] mainly focus on effective mining of topics and regions while ignoring the training efficiency. As mentioned in Section 1.1, training geographical topic models is time-consuming, and thus the existing work only learned small models on small datasets. In this research problem, we aim to develop an efficient distributed solution for training geographical topic models on large datasets. Although there exist several studies on distributed training of topic models [WBS⁺09, SN10, AAG⁺12, LARS14, WZS⁺14, YGH⁺15], no existing distributed solution is developed for training geographical topic models to the best of our knowledge.

Geographical topic models normally have two big matrices of global parameters, i.e., the word-topic distribution matrix and topic-region distribution matrix. Collapsed Gibbs sampling is a popular technique to estimate the parameters of geographical topic models in an iterative manner. To develop a distributed solution for training geographical topic models, we consider the following objectives: support parallelism, reduce the memory requirement, and reduce the communication cost. However, it is challenging to design a solution to achieve these objectives. First, the Gibbs sampling algorithm requires both the word-topic matrix parameters and the topic-region matrix parameters to learn the topic and region for each word token, and thus all the parameters has to be fetched at the beginning of each iteration. Such requirement makes it difficult to parallelize the computation and network communication, and also incur expensive requirement on memory. Second, the global parameters are often distributed to the worker machines in the parameter server implementation. To reduce the communication cost of each worker machine, ideally we want to achieve parameter localization, i.e., each worker mainly works on the part of parameters stored in its local memory, and fetches other parameters in an on-demand manner. However, both region and topic for each word token are unknown before training, it is difficult to achieve parameter localization and fetching parameters on-demand. In addition, another problem is how to distribute the workload to facilitate the distributed learning.

Annotating POIs with geo-tagged microblogs

Most of the recent studies [YCM⁺13, YZS⁺15, WYC⁺15, YCZ⁺15] learn individual user's topic interests from check-in data to support applications like POI recommendation, location prediction, etc. However, as check-in data often contains limited textual

information, we aim to find out microblogs that are semantically related to POIs and associate them to the corresponding POIs. The microblogs associated to POIs can be viewed as a complementary check-in source for mining individual’s topic interests. In addition, such “data integration” benefits map systems like Google Maps in supporting user exploration over real-time information on what happens in the surrounding of a POI. Without loss of generality, we focus on Twitter in this thesis, which is one of the largest microblogging platform.

There exist some studies [LX11, SSSH13, BKB15] on associating geo-tagged tweets to points of interest. They assume that each geo-tagged tweet is associated to a POI. However, we observe that not all geo-tagged tweets are necessarily POI-related in terms of its semantic meaning. In our data collected from Twitter, only 8%-10% of geo-tagged tweets are semantically related to POIs. To the best of our knowledge, no previous work has studied the problem by considering the *semantic relatedness* between geo-tagged tweets and POI.

Determining whether a tweet is semantically related to a POI is challenging, given the volume and shortness of tweets, and the large number of POIs at fine-grained level. Because the coordinates provided by GPS devices are often not precise, the nearest POI is often not the true POI where the tweet was posted. Only 51% and 17% POI-related tweets were posted nearest to their POIs in our Singapore and New York datasets, respectively. In addition, even if a tweet was posted right at a particular POI, it remains hard to determine if the content of the tweet is relevant to the POI. Note that this problem is different from the problem of inferring tweet location, e.g., the coordinates of tweets [WB11, HAG⁺12, AHS13] or the city level location of tweets [DKP12, LHFT14]. These studies assume that the coordinates of tweets are unknown and aim at inferring the missing coordinates for tweets. They complement our research as we target on the geo-tagged tweets.

Mining topical aspects and regions of individuals

As mentioned in Section 1.1, existing models for mining topics of individual users from geo-textual data cannot capture the topical aspects of POIs (e.g., service, environment) and the sentiment of the users. To address these limitations, we aim to explore the interdependency among three latent factors, i.e., topical-aspects, topical-regions, and sentiments, and develop a unified model for capturing the interdependency. Modeling the interdependency in a unified model has the following benefits. Firstly, the learning of topical aspects and topical regions may benefit from each other. For example, some words in the text are related to functional and spatial information of the region, e.g., “restaurant”, “New York”, etc. These words are more likely to appear in topical regions but less likely to appear in topical aspects; Secondly, we can apply the model to many applications, such as POI recommendation, user recommendation and aspect satisfaction analysis in regions. That is, our model can answer questions like:

- Which aspect of a location is favored by people and which is not?
- Which POI would a user like to visit?
- Who would be interested in a given POI?
- What is the overall positive aspect of POIs in the same category in each region?

Building such a unified model is a challenging task. First, the interaction among the three types of latent variables (i.e., aspect, sentiment and region) and the category of POIs is unknown. Second, aspect and sentiment are usually modeled at sentence level [JO11] while region is modeled at the review or document level [YCH⁺11, HAG⁺12], which makes it difficult to estimate the parameters of a unified model.

1.3 Approach and Methodology

Mining and exploring topics in regions

In order to support topic exploration in user specified region and timespan, we propose a query for Topic Mining over documents within a specified Region and a Timespan (TMRT). The TMRT query consists of a rectangular region and a time interval. The answer comprises K topics, each of which is a word distribution. We propose the first framework to efficiently answer the TMRT query. The high-level idea is that, we divide the collection of documents into three-dimensional cells (latitude, longitude and time) and train topic models for the documents within each cell. Then, to compute the topic models for a user specified region and timespan, we combine the topic models of the cells that are located in the specified region and timespan. To realize the idea, we propose several novel techniques. Firstly, we propose a novel and efficient algorithm with a bounded error to combine two LDA topic models learned from two document sets, both falling in the specified region and time period. Different from the existing topic modeling algorithms, our algorithm samples topics for sets of words instead of individual words. The complexity of training the LDA model on two document sets is $O((|W_1| + |W_2|)KI)$, where W_1 and W_2 are sets of word tokens in the two document sets. In contrast, our algorithm reduces the complexity to $O(K(K + |M_2| + |V_2|)I)$, where M_2 is the smaller document set and V_2 is its vocabulary. Their sizes are much smaller than W_1 and W_2 . Secondly, based on the proposed combining algorithm, we develop a new approach to partitioning the collection of spatio-temporal documents into an Octree [Mea82], for indexing the documents. Instead of recursively partitioning the three dimensional space into equal-sized cells, we consider the word overlaps among the generated cells to determine how to perform the partitioning. Our partitioning mechanism enables our combining algorithm to achieve better accuracy, although our proposed combining algorithm works for any

kind of partitioning. We also develop a mechanism to determine on which cells we pre-train topic models to guarantee a given accuracy threshold. Finally, we develop a framework to find the cells whose regions and time periods fall in or overlap with the user specified region and time period. Then we employ our proposed algorithms to combine the pre-trained topic models in these cells to obtain the topics for documents in the specified region and time period.

Distributed training of geographical topic models

To support distributed training of geographical topic models, we propose **PGeoTopic**, the first solution to Parallelize the training of Geographical Topic models over multiple machines. It adopts parameter servers [SN10, HCC⁺13, LAP⁺14] to manage the global parameters. The workers fetch the global parameters from the parameter servers and learn the topics and regions for the data allocated to them. Aiming to achieve three objectives — increase the parallelism, reduce the memory requirement, and reduce the communication cost, **PGeoTopic** comprises the following novel techniques.

- To increase the parallelism while reducing the memory requirement, we propose a new training algorithm that allows independent access to the word-topic matrix and the topic-region matrix. Our algorithm parallelizes fetching the topic-region matrix from parameter servers and training with the word-topic matrix. We also design a model parallelism for the proposed training algorithm that partitions the word-topic matrix by words and the topic-region matrix by regions. The model parallelism makes it possible to support the parallelism of computation and network communication and allows each worker machine only store one slice of either matrix each time.
- We propose to reduce the communication cost in two aspects: (1) We propose a data parallelism with spatial partitioning to make the data of each worker only related to several regions (but not all regions). Then, we propose a two-step sampling scheme to support on-demand parameter fetching, which fetches only parameters of relevant regions for a worker; (2) Based on the first aspect, we store the global parameters in the shared memory of the related worker. Most of the related parameters can be fetched in the local machine, thus reducing communication cost.
- We also propose a new training algorithm to reduce the amortized complexity of sampling each word token from $O(KR)$ to $O(1)$ with bounded errors, where K and R are the numbers of topics and regions, respectively.

Annotating POIs with geo-tagged microblogs

To tackle the challenges in the first research problem, we propose a novel supervised Bayesian model (**sBM** for short) which explores three factors: textual content, coordinates, and user behaviours (user interests in latent regions). Intuitively, the coordinates

act as spatial filters to exclude POIs that are far from a given tweet; the user behaviors provide the user’s interests in some POIs or the types of the POIs to further narrow the search space, and the textual content helps identify the true POI in the small search space. To build the model, we consider the following intuitions:

- A user may post geo-tagged tweets in some preferred latent regions, e.g., shopping streets, sightseeing areas. Each region contains a set of nearby POIs.
- To visit a POI in a region, a user is likely to consider both its popularity in a region and its distance to the region.
- A tweet may or may not be related to a POI. For POI-related tweets, the location and text are determined by the POI where they were posted. For daily conversational tweets that are not related to any POI, we can imagine that their location and text are determined by some dummy POIs that do not exist in the real world. We introduce a set of dummy POIs to model such tweets.
- In a POI-related tweet, users tend to use words related to the POI other than general words. On the other hand, non-POI-related tweets are more likely to include general words.
- If a tweet is POI-related, it is likely that the tweet is close to the POI where it was posted, and the text of the tweet is related to its POI. Moreover, if a tweet is posted in a region that has many attractions, e.g., sight seeing spots, restaurants, etc., it would be more likely to be POI-related other than tweets posted in regions that have fewer attractions.

We model the generative process of each geo-tagged tweet as follows: a user first selects a region to visit. In the region, the user chooses a POI (could be dummy POI) to visit based on its popularity and distance. The words in the tweet are then generated either from POI-related words or general words. Finally, a POI-relatedness variable for each tweet is generated based on the words, distance, and nearby attractions, to identify whether the tweet is POI-related or not.

The novelty of our model is two-fold. First, we add a relatedness response to each document to determine whether a tweet is POI-related by considering the context (e.g., the number of words that are not related to any POI, the distance from the tweet to its most semantically related POI, and the region in which the tweet was posted). By associating words and regions to the relatedness response, regions and language models are better fit to solve the relatedness problem. Second, we introduce a set of dummy POIs to model the location of non-POI-related tweets. The dummy POIs benefit the model in that we can model the POI-related and non-POI-related tweets in a consistent

manner. Moreover, dummy POIs themselves capture spatial and textual information because each dummy POI has its own language model and coordinates like real POIs.

Mining topical aspects and regions of individuals

Previous studies [YCM⁺13, YZS⁺15, WYC⁺15, YCZ⁺15, ZHSS16] model user interests on topics based on the check-in frequency. Intuitively, the true interests should be reflected by the sentiment of the user expressed in the text. Moreover, it is important to uncover the user’s interests on different topical aspects, e.g., environment, such that we can explain to users why a POI is recommended, e.g., because of the environment or price. Because check-in data often contains inadequate textual information for aspect-level topic extraction, we focus on the geo-tagged review data such as Yelp. To model the aforementioned sentiment and aspects, we propose a unified sentiment-aspect-region (SAR) model which mines user interests on topical aspects, topical regions and categories from geo-tagged reviews based on the following intuitions.

- A user visits POIs in a topical-region because the region is geographically convenient to the user (e.g., close to her activity areas) and its topics (e.g., shopping street, education area, etc.) satisfy the user’s interests. Each user has his/her own preferences on the topical-regions.
- A user rates highly of a POI because he/she likes some aspects of the POI. Such preferences might be indicated in the review. Some users like to check the price range of a restaurant first while others might be more concerned with the environment. Moreover, POIs in different categories may have different aspects of interest.
- A user decides to visit a POI in a region by considering the category, topical-aspects of the POI and the distance to it. For example, users may visit POIs of the restaurant category with good environment, but she may first consider the restaurants nearby.
- When a user writes a review on a POI, he/she will use words for both the aspects of the POI and his/her sentiments about the aspects. The user may also use words for the topical-region of the POI. For example, a review on a shop in Times Square may say: “This shop offers best prices in Times Square.” The reviewer uses “price” for *aspect*, “best” for *sentiment* and “Times Square” for *region*.

We model the generative process of a geo-tagged review as follows: a user first selects a region and a category of POIs (e.g., restaurants) to visit. Based on the region and category, the user then visits a POI by considering its popularity in the category and distance to the region. For each sentence in the review, we assume that it expresses one aspect and the corresponding sentiment on the aspect. The words in each sentence are generated based on the aspect, region and sentiment.

Because the model has parameters in both document-level (related to topical-regions) and sentence-level (related to topical-aspects and sentiments), we propose a two-level expectation-maximization learning algorithm. We estimate the document-level parameters in the first step and the sentence-level parameters in the second step.

Compared to the existing user behavior models [YCM⁺13, YZS⁺15, WYC⁺15, YCZ⁺15], our model is novel in that it captures several factors, i.e., sentiment, aspect, region and category, thus improving POI recommendation accuracy. We also propose an efficient online recommendation algorithm based on the proposed model. Apart from POI recommendations, our model also supports explanation of the recommendations, and other applications such as user recommendation and aspect satisfaction analysis in regions.

1.4 Summary of Contributions

Firstly, we define and study the problem of topic exploration on geo-textual documents. To support efficient online querying, we propose a novel and efficient combining algorithm to combine two pre-trained topic models falling in the query region and time span with bounded errors. Our combining algorithm samples the topics for sets of words instead of individual words, thus reducing the complexity. Based on the combining algorithm, we develop a framework with (1) an approach to partitioning the documents into an Octree [Mea82] and pre-training the topic models in the cells of the Octree; and (2) an algorithm to answer the TMRT queries online. We conduct extensive experiments to show that our proposed topic mining framework is able to improve on the runtime by at least an order of magnitude compared to LDA [BNJ03] and a variation of LDA [ABD08] that supports online updates (Online-LDA). Meanwhile, the effectiveness of the proposed topic extraction framework, measured by perplexity, is close to those of LDA and Online-LDA. We also use some example topics extracted by our framework to show that the quality of the extracted topics is comparable to those of LDA and Online-LDA.

Secondly, we propose the first distributed solution for training geographical topic models. The solution comprises a new training method and model parallelism to support the parallelism of computation and network communication while reducing the memory requirement of worker machines. In addition, we propose new techniques for parameter localization to reduce the network communication cost, and a method to reduce the training complexity. We conduct extensive experiments to show that our proposed solution is efficient and scalable in distributed learning of geographical topic models. The experimental results also show that our solution outperforms two baseline solutions by orders of magnitude.

Thirdly, we define the problem of associating POIs with geo-tagged microblogs. We propose a supervised Bayesian model which exploits three factors: textual content, coordinates, and user behaviors under supervision information (e.g., the POI relatedness

variable). We conduct extensive experiments on real-world datasets to demonstrate the effectiveness of the proposed model.

Finally, we propose a novel probabilistic model to capture the interaction of aspect, sentiment, region and category and a corresponding training algorithm. We apply the proposed model to recommend POIs and users and analyze aspect satisfaction in regions. We also propose an algorithm for efficient online POI recommendation. We evaluate our model on real-world datasets to show its effectiveness on POI recommendation and user recommendation. We also demonstrate that our model can explain the reason of the recommendations.

Chapter 2

Literature Survey

My research is related to topic modeling and geographical topic modeling. In this chapter, we first introduce the basic idea of topic modeling and geographical topic modeling, and recent research on distributed training of topic models which is related to our work on distributed training of geographical topic models. Then, we review three types of related studies — (1) location identification of documents related to the work on annotating POIs with microblogs; (2) user behavior modeling related to mining topical aspects and regions of individuals; and (3) online data analytics & exploration related to the problem of topic mining and exploration in regions.

2.1 Topic Models & Geographical Topic Models

2.1.1 Topic Models

Topic models are probabilistic models for discovering “topics” in a collection of documents. Each document may contain several topics, e.g., “sports”, “digital devices”, etc. If a document is more likely to be a “sports” document, words such as “football”, “exercise”, etc., would be more likely to appear in the document than “firework” or “pizza”, etc. Topic models aim to reveal such latent semantic structure of a collection of documents. In topic models, a document is modeled as a mixture of topics and each topic is represented by a multinomial distribution over words.

Latent Dirichlet Allocation (LDA) [BNJ03] is arguably the most successful topic model to date. LDA is a generative model which generates topic distribution θ_d for each document d from a Dirichlet distribution $Dir(\alpha)$ and the word distribution ϕ for each topic from a Dirichlet distribution $Dir(\beta)$. The parameters α and β are hyperparameters of LDA that are used for modeling the sparsity. For each word token v in each document d , LDA draws a topic k from θ_d and then generates v from the corresponding word distribution ϕ_k .

Estimating the parameters, i.e., word distribution for topics ϕ , in LDA is intractable, and several algorithms (e.g., variational Expectation-Maximization algorithm, collapsed Gibbs sampling, etc.) have been proposed to solve the problem. Collapsed Gibbs sampling is a commonly used method to estimate the parameters for LDA. It works as follows: 1) Randomly assign a topic (from a given number of topics K) to each word token in the document; 2) Iteratively update the topic assignment for each word token according to Equation (2.1).

$$p(z^{(m,n)} = k | Z^{-(m,n)}, \alpha, \beta) \propto (c_{m,*}^{k,-(m,n)} + \alpha) \times \frac{c_{*,v}^{k,-(m,n)} + \beta}{\sum_r c_{*,r}^{k,-(m,n)} + |V|\beta}, \quad (2.1)$$

where $z^{(m,n)}$ denotes the topic assignment for the n -th word in document m , $Z^{-(m,n)}$ denotes the topic assignments for word tokens excluding the n -th word token in document m , $c_{m,v}^{k,-(m,n)}$ is the count of assigning topic k to word v in document m excluding the topic assignment of the n -th word token in document m , and wildcard $*$ stands for any document/word. The set V is the set of unique words in the collection of documents which is also called vocabulary. Intuitively, the first part of the equation tells how prevalent is topic k in document m , while the second part tells how prevalent is word v belonging to topic k across all documents. Since Gibbs sampling updates the topic assignment for each word token (m, n) by computing Equation (2.1) for K topics in each iteration, the complexity of the algorithm is $O(|W|KI)$, where W is the set of word tokens in the collection of documents, and I is the number of iterations. It could be very slow when the collection of documents is large. After Gibbs sampling, the language model of a topic, i.e., posterior word distribution given a topic, can be estimated as:

$$p(v|k) = \frac{c_{*,v}^k + \beta}{\sum_{r \in V} c_{*,r}^k + |V|\beta}, \quad (2.2)$$

where $c_{*,v}^k$ is the count of assigning topic k to word v in all documents.

Based on LDA, several online learning algorithms are proposed for text streams [ABD08, YMM09] to speed up the learning process. They aim to sample topic assignments for new documents using the topic model learned from historical data and combine the topic assignments for new documents to construct a new topic model. Sumait et al. [ABD08] propose an online updating method (Online-LDA) to sample new documents using the counts of topic assignments in the historical data ($c_{old,*,v}^k$) as a part of the prior distribution, i.e., we set $\beta' = c_{old,*,v}^k + \beta$. Let $c_{new,m,v}^{k,-(m,n)}$ be the count of assigning topic k to word v in the m -th new document, excluding the topic assignment for the n -th word token in

the m -th new document. The update function for online-LDA is:

$$p(z_{new}^{(m,n)} = k | Z_{new}^{-(m,n)}, Z_{old}, \alpha, \beta) \propto (c_{new,m,*}^{k,-(m,n)} + \alpha) \times \frac{c_{new,*,v}^{k,-(m,n)} + c_{old,*,v}^k + \beta}{\sum_r c_{new,*,r}^{k,-(m,n)} + c_{old,*,r}^k + |V|\beta}, \quad (2.3)$$

where $Z_{new}^{-(m,n)}$ is the topic assignments to word tokens in the new document collection excluding the n -th word in document m , and Z_{old} is the topic assignments to word tokens in the historical data.

Online-LDA has the same complexity as LDA on the set of new documents, i.e., $O(|W_{new}|KI)$, where W_{new} is the set of word tokens in the new documents. But online-LDA saves the computation for sampling topics on the old documents. Online-LDA is the state-of-the-art technique which can be used to learn topics in a given region and timespan. However, to the best of our knowledge, no existing work proposes techniques for combining two LDA topic models pre-trained from two document sets into one model to support mining and exploring topics in a region and a time span.

2.1.2 Geographical Topic Models

Geographical topic models extend topic models in order to find geographical topics and topical regions. They have been extensively studied recently [Siz10, YCH⁺11, HAG⁺12, LFYX13, AHS13, HJE13, YSC⁺13, YCM⁺13, LXP⁺15, ZCYZ15, LEHC15, GG16]. They are proven to be effective in many applications, such as point-of-interest recommendation [YSC⁺13, LFYX13, LXP⁺15, LEHC15, ZCYZ15], event detection [GG16], user mobility modeling [YCM⁺13], etc. GeoFolk [Siz10] attempts to model the spatial aspect of topics, and each region in GeoFolk is associated to one latent topic. Yin et al. [YCH⁺11] break the one to one relation between region and topic and define a topic distribution of each region. To this end, the relation between topic and region forms a topic-region matrix. Recent studies [YCM⁺13, AHS13, GG16] generally follow this formulation, while there also exist some matrix factorization based models ignoring the relation between topics and regions [LFYX13, LXP⁺15]. Example 1 shows a geographical topic model follows Yin et al.'s formulation [YCH⁺11].

Example 1: Let $D = \{d_1, d_2, \dots, d_{10}\}$ be a set of geo-tagged documents. Figure 2.1 illustrates the location of each document. The word tokens of each document are listed in Table 2.1. Suppose we want to mine two topics and two regions, respectively. The geographical topic model assigns a topic label (k_1 or k_2) for each word token as listed in Table 2.1 and returns the topical regions r_1 and r_2 in Figure 2.1. Each region is mainly about some of the topics, e.g., region r_1 is mainly about the movie topic, and is often represented by a continuous distribution (e.g., Gaussian distribution). Each topic

is described by a distribution of words. In the example, the movie topic is represented by the distribution $\langle \text{“movie”} = \frac{1}{8}, \text{“cold”} = \frac{1}{8}, \text{“3D”} = \frac{1}{16}, \dots \rangle$. The topic distribution of each region is computed by counting the topic assignments to word tokens in the regions. For example, the region r_1 is $\frac{11}{15}$ related to “movie”, and $\frac{4}{15}$ related to “food”.

Figure 2.1: An example of geographical topic model

Table 2.1: Word tokens of documents. The topic estimated for each word token is placed in the brackets after the word token. k_1 and k_2 stand for the movie topic and food topic, respectively.

doc	word tokens
d_1	wonder woman (k_1), ticket (k_1)
d_2	Atmos (k_1), Dolby (k_1), movie (k_1)
d_3	chicken (k_2), soup (k_2), sauce (k_2), taste (k_2)
d_4	3D (k_1), movie (k_1), late (k_1)
d_5	pirates (k_1), Caribbean (k_1), cold (k_1)
d_6	terrible (k_2), service (k_2), cold (k_1)
d_7	seat (k_2), sci-fiction (k_1), boss (k_1)
d_8	snack (k_2), seafood (k_2), crab (k_2), service (k_2)
d_9	promotion (k_1), popcorn (k_1), show (k_1)
d_{10}	seat (k_2), noodle (k_2), korean (k_2), kimchi (k_2)

In most of the existing models [YCH⁺11, AHS13, YCM⁺13], each latent topic is modeled as a multinomial distribution of words, and each latent region r is often modeled by a Gaussian distribution $\mathcal{N}(\mu_r, \Sigma_r)$ with mean μ_r and covariance Σ_r . The correlation between topic and region can be formulated in different ways according to different purposes of the models. Some models [YCH⁺11, YCM⁺13] such as Example 1 tend to capture

Figure 2.2: Graphical representation of a typical geographical topic model

the topic distribution in topical regions, while others [GG16] aim to reveal the region distribution of a topic. In the models to discover the topic distribution of regions, the topics follow the distribution $p(z|r) = \frac{m_{zr}}{\sum_{z'} m_{z'r}}$, where m_{zr} is the frequency of assigning topic z to region r . Figure 2.2 shows the graphical representation of such geographical topic model. In the graphical representation, D is the set of documents, and N_d is the number of word tokens in document d . The model generates a document according to the following process:

1. A latent region r is first randomly sampled from a global Dirichlet distribution $\text{Dir}(\alpha)$;
2. The topic of each word token is then generated from the topic distribution of r , i.e., $p(z|r) = \frac{m_{zr}}{\sum_{z'} m_{z'r}}$;
3. Finally, each word is generated from its topic according to the topic's word distribution ϕ , while the coordinates of each document are generated from the Gaussian distribution of region r .

For the other type [GG16] that captures the region distributions of topics, regions are generated conditioning on topics in the second step of the aforementioned generative process according to the probability $p(r|z) = \frac{m_{zr}}{\sum_{r'} m_{zr'}}$. Because the two types of models are similar except the modeling of the topic-region relation, our parallel training algorithm is applicable to both types. For simplicity, we focus on the first type of models in this thesis.

The goal of training a geographical topic model is to estimate the posterior distribution of topic and region for each word token given the data, so that we can sample and assign the topic for each word token and the region for each document as in Table 2.1 and Figure 2.1 according to the posterior. The region assignment of the document

can then be decided by uniformly choosing a region assignment from the word tokens. However, the posterior does not have a close form solution. Collapsed Gibbs sampling is a popular technique for estimating the posterior. It draws the current sample (e.g., topic and region) based on the previous samples. Specifically, a region r and a topic k is sampled for the i -th word token w in the document d using the following posterior distribution of topic and region, given the topic and region assignments of other word tokens (Φ^{-di}):

$$\begin{aligned}
 p(z^{(m,n)} = k, r^{(m,n)} = r | \Phi^{-(m,n)}) \propto & \underbrace{\frac{c_{*,v}^{k,-(m,n)} + \beta_{kv}}{\sum_{v'} c_{*,v'}^{k,-(m,n)} + \beta_{kv'}}}_{\text{word-topic matrix}} \cdot \underbrace{\frac{m_{kr}^{-(m,n)} + \gamma_{kr}}{\sum_{k'} m_{k'r}^{-(m,n)} + \gamma_{k'r}}}_{\text{topic-region matrix}} \\
 & \cdot \underbrace{f(l_d | \mu_r, \Sigma_r) \cdot (c_r^{-m} + \alpha_r)}_{\text{location-region relation}},
 \end{aligned} \tag{2.4}$$

where $c_{*,v}^{k,-(m,n)}$ denotes the frequency of assigning word v to topic k without counting the assignment of the current n -th word token in document m . The notation $m_{kr}^{-(m,n)}$ denotes the frequency of assigning topic k to region r excluding the assignment of the current token. The hyperparameters α_r , β_{kv} , γ_{kr} are the prior probability of region r , the prior probability of observing topic k together with word w , and the prior probability of observing topic k together with region r , respectively¹. The function $f(\cdot | \mu_r, \Sigma_r)$ denotes the density function of Gaussian distribution over coordinates in region r . The count c_r^{-m} is the frequency of region r excluding document d .

The training process runs iteratively until the model converges, and in each iteration, Equation (2.4) is used to sample the region and topic for all word tokens in the dataset. In general, Equation (2.4) consists of three terms. The first term is the same as the word-topic relation in topic models (e.g., LDA) which represents the word distribution of a topic without considering the topic assignment of the current token. The second term is topic-region correlation matrix without counting the assignment of current token. The third term is the Gaussian distribution of region multiplies the region distribution in the dataset. The first two terms can be stored as two-dimensional matrices. For simplicity, we use **word-topic matrix** and **topic-region matrix** to refer to the two parameters in the remaining of the thesis. The Gibbs sampler needs to access the whole topic-region matrix when computing Equation (2.4) for all topic-region pairs. When the topic-region matrix is large, it is difficult to be synchronized among machines in a distributed setting and may not even fit into the local memory.

It is not difficult to observe that randomly sampling a topic and a region for each word token takes $O(KR)$ time, because it takes $O(KR)$ to compute the joint probabilities

¹These prior distributions are often set to uniform when we do not have any prior knowledge of the distribution.

of all the topic-region pairs, and takes $O(KR)$ time to sample a topic-region pair from $K \times R$ possible values. The total complexity of training a geographical topic model on a dataset of word tokens W is $O(KR|W|I)$, where I is the number of iterations. In practice, training a model with thousands of topics and regions on a dataset with millions of documents could take months.

2.1.3 Distributed Training of Topic Models

Distributed training of the LDA model is most related to our work on distributed training of geographical topic models. It gains plenty attractions in recent studies [NASW09, AAG⁺12, LZCS11, ZR13, WZS⁺14, YGH⁺15, YHY⁺15, CLZC16]. HD-LDA [NASW09] distributes data across different processors, and the processors perform Gibbs Sampling on local data in each iteration, followed by a global synchronization of parameters. Ahmed et al. [AAG⁺12] and Li et al. [LZCS11] apply data-parallel strategy, in which they partition and distribute the document collection to the workers and the model, i.e., the word-topic distributions, is stored in the shared memory across machines (called parameter servers). The data is not moved around because it is often larger than the model. Instead, the model is transmitted to the workers and each worker only learns topics over local data. However, when the model cannot be fit into memory, parts of the model are frequently swapped in and out of the worker's main memory, leading to expensive I/O cost.

Apart from data-parallelism, model-parallelism of LDA has been proposed in some existing studies. F+LDA [YHY⁺15] partitions the data across multiple workers and builds a binary tree structure to speed up the sampling. Peacock [WZS⁺14] and LightLDA [YGH⁺15] adopt both data parallel and model parallel strategies. Peacock [WZS⁺14] partitions the model by words and each worker takes care of the words appearing in its model partition. Although Peacock parallelizes the model, it needs to synchronize the extra document-topic distributions because a document is processed across workers. LightLDA partitions the model into slices by words as in Peacock to achieve model parallelism, but distributes each document to a single worker for data parallelism. For efficient training, LightLDA schedules the model slices to workers such that the workers process different parts of the model independently at the same time. After all the word tokens related to the model slice in a worker are processed, LightLDA schedules another model slice to the worker until all the model slices are processed by each worker in each iteration. The updates of the parameters are synchronized after every s iterations, where s is the staleness parameter in stale-synchronous parameter servers [HCC⁺13]. WarpLDA [CLZC16] further enhances the performance of LightLDA by a carefully designed memory access mechanism.

The aforementioned techniques are not suitable to geographical topic models. For example, if we extend LightLDA or WarpLDA, the region-topic matrix will be accessed

each time a worker processes a model slice of the word-topic matrix. The more model slices of the word-topic matrix we have, the more access to the region-topic matrix occurs. Worse still, the region-topic matrix may not be able to fit into memory. To the best of our knowledge, no existing work considers training a geographical topic model in a distributed setting. In this thesis, we propose the first distributed solution of training geographical topic models to support parallelism, reduce local memory requirement, and reduce network communication cost.

2.2 Online Data Analytics & Exploration

2.2.1 Data Cube & Topic Cube

Our work on topic mining and exploration in regions is inspired by the concept of data cube [HK00], which is widely used for Online Analytical Processing (OLAP) on multi-dimensional data with operations like slice, dice, roll-up, and drill-down. Each row in the multi-dimensional text database contains some structured dimensions (i.e., attributes) and unstructured text data. The existing work on analyzing and exploring multi-dimensional text database is closest to our problem.

Lin et al. [LDH⁺08] propose a text-cube model on multi-dimensional text database. Text cube is a data cube where each cell aggregates a set of documents with matching attribute values in a subset of dimensions. Based on the text cube, they study the problem of processing an IR query (i.e., a set of terms) with constraints on dimensions to retrieve relevant documents. Simitsis et al. [SBSR08] propose a keyword driven OLAP system over a collection of multidimensional text data. Operations of roll-up and drill-down can be performed efficiently and accurately based on the content and the link-structure of a dynamically selected document subset during the query time. In addition, Ding et al. [DZL⁺11] study the problem of finding top- k most relevant cells in text cube for a keyword query. Similarly, based on the text cube, Zhao et al. [ZLDH11] develop the TEXplorer system to rank candidate dimensions and cells for a keyword query.

However, those text-cube based proposals do not consider the semantic meaning (i.e., topics) of each term in documents. They are actually orthogonal to our work, and our proposed techniques can be integrated into these proposals for exploring spatial-temporal topics of data cube. Moreover, our proposed combining algorithm fits perfectly to support the roll-up operation for topics of a text cube.

Zhang et al. [ZZH09] develop a topic-cube model for a multidimensional text database. Specifically, the topic cube can be viewed as a standard data cube augmented by a topic dimension based on a hierarchical topic tree and the objective of the topic-cube is to support users to drill-down and roll-up the text dimension along the topic hierarchy. The topic tree in [ZZH09] is based on the topic model generated from all documents in

the dataset. However, our problem requires the topic model to be generated from the documents in the user specified spatial region and time interval. Moreover, topic-cube does not support mining topics for user specified regions and timespans. Therefore, topic cube cannot be applied to our problem.

2.2.2 Content Exploration

Some recent work aims at developing online data exploration mechanism based on clustering methods. Zhang et al. [ZZH11] propose MiTexCube by augmenting each cell with micro-clusters that make the online processing more efficient. More recently, Feng et al. [FZZ⁺15] propose a system called StreamCube to explore events over the spatio-temporal Twitter stream by clustering hashtags. Each hashtag in StreamCube is represented by a vector of terms/hashtags that co-occurred with the hashtag. Consequently, the clusters are generated based on the tweets that have hashtag(s), which only take 11% tweets according to a statistical study [HCC11]. One can use keyword extracted from the text instead of hashtags to address the sparsity problem in StreamCube. However, we focus on the problem that efficiently mines topics of a collection of documents in the user specified region and timespan, as well as the topic assignment of each selected document. The comparison of clustering and topic modeling techniques is not our focus and we leave this as future work.

Angel et al. [AKSS09] propose the Grapevine system that aims to track entities in a large scale of documents. Grapevine extracts real-word entities (e.g., Barack Obama) from a collection of text data (e.g., blog posts, news articles, and tweets) using available entity extraction algorithms (e.g., [SAKS10]). Given a temporal and demographic restriction (e.g., location, gender), the topic in Grapevine is defined by the most talked-about entities for the chosen dimensions. However, our topics are discovered from topic models where each topic is represented by a probabilistic distribution over words. In addition, Grapevine uses semantic locations (country, state, city), while we consider geospatial coordinates.

2.3 Location Identification of Documents

We next survey the studies on location identification which relate to our work on annotating POIs with microblogs. We categorize these studies into the following two types.

Associating POIs to GPS records

The first type of studies aims to associate a POI to a GPS record or other post with geospatial coordinates [LX11, LX14, NTKS14]. Lian et al. [LX11] extract several features between a GPS record and a POI and apply them in a learning-to-rank model, to select the most appropriate POI for the GPS record. The extracted features include

the popularity of the POI, distance between the GPS record and the POI, frequently check-in time slots of the POI, and the number of user check-ins at the POI. The authors further extend their method [LX14] by factorizing the user-POI check-in matrix and the time-POI check-in matrix, and use the resulting latent vector of user, POI and time as extra features in the learning-to-rank model. Nishida et al. [NTKS14] propose to extract stay points from user trajectories and then assign a POI to each stay point by considering the spatio-temporal information of the stay point. Our problem of annotating POIs with microblogs is different from this type of studies in that we need to identify the POI-related tweets from those that are not related. That is, we consider the content of the tweets.

Associating POIs to documents

The second type of studies associate a POI to plain text, e.g., tweets or other social media posts without geo-tags [KMO11, DKP12, LHFT14, LPC⁺15]. Dalvi et al. [DKP12] propose a probabilistic model to infer the user’s location and match tweets to spatial objects such as restaurants. They assume that each user has a location which can be inferred from the user’s visit history. They then use the inferred user location together with a language model to identify the spatial object for a tweet. However, a user may travel to many locations and have multiple activity regions in a city [CML11]. If a tweet is posted at a “Starbucks” that is far from the inferred user location, their model probably matches the tweet to a “Starbucks” near the inferred user location. Moreover, as reported in their paper, it is difficult to infer the accurate user location. A radius of 10 miles is therefore used to represent user’s location, which is too far to help in identifying POIs in dense regions. Kinsella et al. [KMO11] propose to identify spatial objects for each tweet at different granularity levels, from country to zip code. They model each spatial object using a language model and then compare with the language model of the tweet to locate the most probable spatial object. Li et al. [LHFT14] compute a coarse-grained user location at city level and aim to disambiguate POIs with the same name but in different cities. Li et al. [LPC⁺15] propose to combine the visual features and textual features to infer the POI for an Instagram photo. This type of studies assumes that tweets are not geo-tagged, and aim at inferring the coordinate locations of tweets. In contrast, we consider geo-tagged tweets and use them to annotate POIs. In fact, these proposals are complementary to our work for tweets without geo-tags.

2.4 User Mobility & Interest Modeling

2.4.1 User Behaviour Models

Many efforts have been done in modeling user behaviors in geographical data [HAG⁺12, KIH⁺13, HE13, LFYX13, YCM⁺13, YCC⁺14a, YCC⁺14b, ZCYZ15]. Different from

the general type of geographical topic models [Siz10, YCH⁺11, AHS13] which aim to discover and compare topics in different regions, these models focus on mining user interests on different regions and different topics to help POI recommendation, location prediction, etc. Hong et al. [HAG⁺12] and Hu et al. [HE13] study user preference on geographical topics and regions in geo-tagged tweets. Kurashima et al. [KIH⁺13] extend the LDA model by considering the user’s locations from historical check-ins to learn user interests on topics. Yuan et al. [YCM⁺13] explore personalized regions for individuals and incorporate temporal information in modeling user behaviors. Yin et al. [YCC⁺14a, YCC⁺14b] consider user interests over time.

The existing user behavior models mainly focus on mining general topics, e.g., restaurants, entertainment, etc., while ignoring the user interests on topical aspects such as taste, price, etc. Such fine-grained preference is essential in accurate recommendations and predictions. What’s more, the existing methods does not consider the user’s sentiment expressed in the check-ins or social media posts, leading to misunderstanding of user interests. A user may hold negative opinions on some topic even though they mention the topic many times. In this thesis, we study a different problem of mining topical aspects and regions of individual with consideration of sentiment.

Our work on annotating POIs with microblogs also takes user behavior into account, but is different from the existing user behavior models. Most of the existing proposals for user behavior modeling use Foursquare check-ins data, and they assume that each post is related to POIs. In contrast, we consider both POI related and unrelated tweets. Because the existing user behaviour models always associate a tweet with a POI, it is not applicable to our problem. Moreover, our proposed model introduces a relatedness response and a set of dummy POIs to better fit the POI annotation problem, which have not been explored in existing models.

2.4.2 Sentiment Analysis

Our work on mining topical aspects and regions of individuals is also related to the work on sentiment aspect joint modeling. We divide the existing studies into two types: sentence level and phrase level.

Sentence level sentiment analysis supposes each sentence expresses one aspect of the product. Titov and McDonald [TM08a] present a sentence level model, namely MG-LDA, to extract aspects from reviews. Based on MG-LDA, Titov and McDonald further propose a method [TM08b] which jointly models the aspect and rating. The model is then used to predict aspect ratings from a review. This model can extract aspect level sentiment/rating, but needs aspect rating as observable variable, which is often not available. Jo and Oh [JO11] model aspect and sentiment jointly in an LDA-based framework. Both aspect and sentiment are modeled as latent variables in the model,

and each word has a joint distribution over topics and sentiment polarities. To identify the polarities, they keep a list of seed words for each sentiment polarity, and give higher probability to generate a seed word from its corresponding polarity.

Studies on phrase level sentiment [MLW⁺07, WLZ10, LCDZ11, ME11, ZZZ⁺14] use NLP tools to analyze the dependency between the words in a sentence and extract aspect-opinion phrase pairs, e.g., $\langle \textit{screen}, \textit{bright} \rangle$, with some predefined patterns. The aspect-opinion phrase pairs are then used for further analysis on the sentiment polarities on the aspects. Mei et al. [MLW⁺07] build a PLSA model in which a topic is modeled as a linear mixture of multinomials from neutral topics and two sentiments (positive and negative). Wang et al. [WLZ10] build a regression model to capture the aspect ratings from the overall ratings. The overall ratings are modeled as the weighted sum of the sentiments from all aspects. However, aspects are fixed and extracted using a list of seed words and a boot-strapping algorithm. Moghaddam and Ester [ME11] propose an interdependent LDA model in which the aspects of a product is modeled and the corresponding ratings to each aspect is drawn depending on the aspect. However, the ratings are treated as 5 clusters and the model cannot tell which cluster has the rating of 1 and which has the rating of 5.

Different from these methods, we jointly consider region, aspect, category and sentiment in a unified model to support many applications, e.g., POI recommendation, user recommendation, aspect satisfaction analysis, etc.

Chapter 3

Mining and Exploring Topics in Regions

As large collections of geo-textual documents are publicly available on the internet, it is interesting to provide an efficient solution for users to explore such geo-textual documents because the data is often too “big” for the users to consume. In this chapter, we focus on the problem of topic mining and exploration in user specified regions and timespans which provides users insights about events, trends, and public concerns within the specified region and timespan. Different from the existing work on data warehouse [HK00, LDH⁺08, DZL⁺11, ZLDH11, ZZH09] that only allows users to slice, dice, roll-up, and drill-down at predefined cells, we aim to support querying topics in any user specified region and timespan in an exploratory manner. We propose the first framework that (1) divides the documents into cells (according to latitude, longitude and time) and pre-trains topic models on the cells; and then (2) combine the pre-trained models fall in the query region and timespan. We show that the proposed framework is efficient and achieves at least an order of magnitude faster than baseline methods in experiments.

This chapter is organized as follows. Section 3.1 defines the problem of mining topics within a specified region and a timespan. Section 3.2 provides an overview of the proposed framework for solving the problem. Section 3.3 gives details of each component of the framework, including the online combining algorithm of two pre-trained models and the off-line pre-computation. Section 3.4 reports the experimental results and Section 3.5 conclude this chapter.

3.1 Problem Definition

We introduce our data model and define the problem of mining topics over a specified region and timespan.

3.1.1 Data Model

Definition 1: Spatio-Temporal Document. A spatio-temporal document is denoted by a quadruple $d = \langle id, l, t_c, w \rangle$, where id is the document id, which is assigned based on t_c , the creation time of d , l is a location with latitude and longitude, and w is a sequence of word tokens from the vocabulary $V = \{v_1, v_2, \dots, v_{|V|}\}$. We use $d.w[i]$ to denote the i -th token in $d.w$. $\square$

The spatio-temporal documents in Definition 1 can be geo-tagged tweets in Twitter, geo-tagged news in news portals, geo-tagged photos with tags in Flickr, Points of Interest (POIs) and check-ins with text descriptions in Foursquare, etc. Without loss of generality, we consider each document has only one location. When a document contains more than one location, we duplicate the documents for each location. For example, if a tweet about European SuperCup is related to three locations, (e.g., Barcelona, Sevilla, and Tblisi), we generate three copies of the tweet and each copy is assigned to one location.

3.1.2 Problem Definition

Definition 2: Topic Mining over Documents within a Specified Region and a Timespan Query (TMRT Query). Let D be a set of spatio-temporal documents, R be a rectangular region, and $[t_b, t_e]$ be a time period. TMRT query is to mine K topics from documents in D whose locations fall in R and creation time falls in $[t_b, t_e]$, as well as the topic assignment of each selected document, based on a given topic model. $\square$

In this chapter, we use LDA to mine topics for spatio-temporal documents because it is one of the most commonly used topic models.

Figure 3.1: An example of TMRT query

Example 2: Let $D = \{d_1, d_2, \dots, d_{12}\}$ be a set of spatio-temporal documents. The location of each document is presented in Figure 3.1. The creation time and the word

Table 3.1: Information of documents

doc	time of creation	tokens
d_1	09:42 Jul 3, 2015	yummy, pizza
d_2	10:18 Jul 6, 2015	espresso, bar, night
d_3	13:43 Jul 4, 2015	Barcelona, champion, excited
d_4	19:22 Jul 7, 2015	football, excited
d_5	17:40 Jul 6, 2015	iPhone, Nike, exercise
d_6	12:23 Jul 8, 2015	sleep, tired, holiday
d_7	22:57 Jul 3, 2015	Barcelona, sightseeing
d_8	18:41 Jul 4, 2015	marathon, exhausted
d_9	21:42 Jul 4, 2015	holiday, parade, independence
d_{10}	23:33 Jul 4, 2015	independence, firework
d_{11}	17:19 Jul 3, 2015	3D, movie, boring
d_{12}	10:40 Jul 7, 2015	sigmod, deadline

Table 3.2: Final topic assignment of R_1

document	topic of tokens
d_2	espresso (k_0), bar (k_1), night (k_1)
d_3	Barcelona (k_2), champion (k_2), excited (k_0)
d_4	football (k_2), excited (k_0)
d_5	iPhone (k_1), Nike (k_2), exercise (k_2)

tokens of each document are presented in Table 3.1. The two rectangles R_1 and R_2 and their corresponding timespans in Figure 3.1 represent two TMRT queries.

Based on the set of documents falling in the corresponding spatial regions and timespans of R_1 and R_2 , which are presented in the first column of Table 3.2 and Table 3.3, respectively, we mine LDA topic models by employing Gibbs sampling algorithm. Suppose the number of topics (i.e., “ K ”) is set to 3, the final topic assignments for each document in R_1 and R_2 are presented in the second column of Table 3.2 and Table 3.3, respectively. The three topics of R_1 are given as follows (suppose the prior of word distribution of each topic $\beta = 0$ in LDA):

$$k_0: \langle p(\text{excited}—k_0)=0.67, p(\text{espresso}—k_0)=0.33 \rangle$$

$$k_1: \langle p(\text{bar}—k_1)=0.33, p(\text{night}—k_1)=0.33, \\ p(\text{iPhone}—k_1)=0.33 \rangle$$

$$k_2: \langle p(\text{Barcelona}—k_2)=0.2, p(\text{champion}—k_2)=0.2, \\ p(\text{football}—k_2)=0.2, p(\text{Nike}—k_2)=0.2, \\ p(\text{exercise}—k_2)=0.2 \rangle.$$

The topics of R_2 is as follows:

$$k_0: \langle p(\text{holiday}—k_0)=0.5, p(\text{iPhone}—k_0)=0.25, \\ p(\text{sleep}—k_0)=0.25 \rangle$$

Table 3.3: Final topic assignment R_2

document	topic of tokens
d_5	iPhone (k_0), Nike (k_1) exercise (k_1)
d_6	sleep (k_0), tired (k_1), holiday (k_0)
d_9	holiday (k_0), parade (k_2), independence (k_2)
d_{10}	independence (k_2), firework (k_2)

$$\begin{aligned}
k_1: & \langle p(\text{Nike}—k_1)=0.33, p(\text{exercise}—k_1)=0.33, \\
& p(\text{tired}—k_1)=0.33) \\
k_2: & \langle p(\text{independence}—k_2)=0.5, p(\text{parade}—k_2)=0.25, \\
& p(\text{firework}—k_2)=0.25).
\end{aligned}$$

□

3.2 Framework Overview

Based on LDA, we aim at mining K topics from a spatio-temporal document collection D for a TMRT query Q , which comprises a user specified region R and time interval $[t_b, t_e]$. In general, the collection of spatio-temporal documents is treated as a data warehouse and we conduct topic mining over a subset of documents falling in the region and time interval specified in Q . A straightforward solution would work as follows: We first find a set of documents D' such that for all $d \in D'$, d locates in R and $d.t_c \in [t_b, t_e]$. Then we learn K topics from the documents in D' using the collapsed Gibbs sampling technique [Liu94]. However, as discussed in Section 2.1.1, it is time-consuming ($O(|W|KI)$) to train the topic model every time a TMRT query is submitted. Hence, we need a more efficient mechanism to handle the TMRT query.

An underlying idea of many approaches to performing OLAP is to organize the data into data cubes with dimensional hierarchies and summary statistics (e.g., [FZZ⁺15, ZZH09, LDH⁺08]). With the help of data cubes, we are able to efficiently explore the data by different granularity levels [LDH⁺08]. Inspired by the idea of data cube, we organize the spatio-temporal documents into cells in a hierarchical index, and pre-compute topic models for a set of selected cells.

The remaining challenges of processing the TMRT query include: (1) How to leverage the pre-trained models associated with indexed cells to efficiently compute the topic model for the given query Q with region R and timespan $[t_b, t_e]$; (2) How to organize the documents into a hierarchical index, and which cells in the hierarchical index do we pre-compute topic models to help accelerate online topic mining for the TMRT query?

To address the first challenge, we propose a novel idea for efficiently combining the topic models of two cells (Section 3.3.1). Our idea is to accelerate the topic learning process by sampling topics for a set of word tokens instead of sampling topics for individual

word tokens. We prove that our combining algorithm has a bounded error compared with the Online-LDA algorithm. Based on our idea of combining topic models, we propose an online topic mining algorithm to search for cells which are completely or highly covered by the spatial region and timespan of query Q , and combine the models of these cells using our combining algorithm (Section 3.3.4).

To address the second challenge, we propose to organize the spatio-temporal documents based on the idea of Octree [Mea82]. Octree is an extension of Quadtree in a three-dimensional space. It recursively partitions the space into eight equal-sized cells. Different from the traditional Octree in which a cell is partitioned into eight equal-sized subcells, we do the partitioning differently – we consider two objectives in determining how to partition a cell: 1) minimizing the word overlap among the subcells; and 2) balancing the number of documents among the subcells (Section 3.3.3). In addition, we propose an algorithm to determine on which cells to pre-train topic models and when to stop partitioning such that the space requirement does not exceed a specified threshold and the error of combining topic models does not exceed a predefined bound.

Figure 3.2 illustrates the framework of our approach. The documents are indexed by an Octree. Given a TMRT query with a rectangle and a timespan, we traverse the Octree to retrieve the topic models of the cells covered by the rectangle and the timespan. Next we combine these retrieved topic models. Finally we output the K topics of the new topic model computed by our combining algorithm.

3.3 Algorithms

In this section, we present the details of our algorithms for handling the TMRT queries. Specifically, we first present our sampling algorithm for combining the topic models of two Octree cells and its extension to combining multiple cells in Section 3.3.1. Then, we present a method to handle the cells that overlap with, but are not contained by the user specified region and timespan (Section 3.3.2). We present our method for partitioning the spatio-temporal documents into an Octree and the offline pre-computation for Octree cells in Section 3.3.3. Finally, we give our online topic learning algorithm for a TMRT query in Section 3.3.4.

3.3.1 Combining Topics of Two Octree Cells

In this subsection, we consider the case that the given region R and time interval $[t_b, t_e]$ in the TMRT problem exactly cover two Octree cells, for which we have pre-trained topic models. We will consider the general case and how to construct the Octrees in the subsequent subsections. We aim to combine the two pre-trained topic models into one topic model within R and $[t_b, t_e]$. The combining is challenging in that the topics in one

Figure 3.2: Framework for TMRT query

model cannot be simply associated to the other model. Example 3 shows the pre-trained topics of two cells, which are represented by distribution over words. Even though k_2 in cell 1 is similar to k_1 in cell 2 (about sports), they are not exactly the same topic because their distributions are different. Moreover, a topic in one cell could be similar to multiple topics in the other cell. For example, k_0 in cell 2 is similar to k_0 and k_1 in cell 1. Due to the incompatibility of topics between the two models, we cannot simply combine them. We need to re-sample the topic models of the cells to make them compatible for combination.

Example 3: Pre-trained topics for cell 1:

$$k_0: \langle p(\text{iPhone}—k_0)=0.43, p(\text{Android}—k_0)=0.57 \rangle$$

$$k_1: \langle p(\text{bar}—k_1)=0.33, p(\text{night}—k_1)=0.33, \\ p(\text{iPhone}—k_1)=0.33 \rangle$$

$$k_2: \langle p(\text{Barcelona}—k_2)=0.25, p(\text{champion}—k_2)=0.25, \\ p(\text{football}—k_2)=0.25, p(\text{Nike}—k_2)=0.25 \rangle$$

Pre-trained topics for cell 2:

$$\begin{aligned}
 k_0: & \langle p(\text{holiday}—k_0)=0.5, p(\text{iPhone}—k_0)=0.25, \\
 & p(\text{sleep}—k_0)=0.25) \\
 k_1: & \langle p(\text{Nike}—k_1)=0.33, p(\text{exercise}—k_1)=0.33, \\
 & p(\text{tired}—k_1)=0.33) \\
 k_2: & \langle p(\text{independence}—k_2)=0.5, p(\text{parade}—k_2)=0.25, \\
 & p(\text{firework}—k_2)=0.25).
 \end{aligned}$$

□

There exist two solutions to learn the topic model for a TMRT query: 1) re-training an LDA model [BNJ03] for the spatio-temporal documents in the two cells; and 2) re-sampling the topics for one of the two Octree cells given the topic assignment of the other one using Online-LDA [ABD08]. However, both solutions suffer from high computation cost. Let the sets of word tokens in the two cells be W_1 and W_2 , respectively, where W_2 is for the cell containing fewer words, i.e., $|W_2| < |W_1|$. The first solution needs to sample topics for all the documents in the two cells, and its complexity is $O((|W_1| + |W_2|)KI)$. The second solution only samples topics for documents in the cell containing fewer word tokens, and thus the training time can be reduced at least by half, i.e., $O(|W_2|KI)$. Although the second solution utilizes the pre-trained model of one cell, it fails to utilize the other pre-trained model.

To this end, we propose a combining algorithm that is able to leverage the pre-trained models of both cells. We reduce the complexity to $O((K + |M_2| + |V_2|)KI)$, where M_2 and V_2 are the set of documents and the vocabulary of the smaller cell, respectively. Note that $K + |M_2| + |V_2|$ is much smaller than $|W_2|$. We also prove that our algorithm has a bounded sampling error compared to Online-LDA.

Similar to Online-LDA, we use the topic assignment of one cell to re-train the model for the other cell. The significant difference is that we do not sample the topic for each word token in the documents one by one as in Online-LDA. Instead, we sample the topic for word tokens that are assigned to the same topic in the pre-trained model simultaneously. Since this treatment is an approximation to Online-LDA, we prove an error bound of our sampling algorithm compared to Online-LDA. To the best of our knowledge, the idea is new although the topic model has attracted extensive research attention.

Let Z_1 and Z_2 be the topic assignment for the two sets of word tokens W_1 and W_2 , respectively. Without loss of generality, we assume that $|W_1| > |W_2|$. We want to use the topic assignments Z_1 of the pre-trained model for cell 1 to resample Z_2 . Consider the fact that the topic assignments of the pre-trained model divide the set of tokens W_2 into K subsets: $\{W_{21}, W_{22}, \dots, W_{2K}\}$. Thus, we can jointly sample the word tokens of each subset according to the joint posterior distribution given the other topic assignments: $p(Z_2^{W_{2i}} = k | Z_2^{-W_{2i}}, Z_1, \alpha, \beta)$. Here, we denote $Z_2^{W_{2i}} = k$ as the event that all word tokens

in W_{2i} are assigned to topic k and $Z_2^{-W_{2i}}$ as the topic assignments for the other subsets. The parameter α is prior of topic distribution and β is prior of word distribution for each topic as used in LDA.

Learning: We use collapsed Gibbs sampling to learn topics for the word tokens. Specifically, we compute the complete likelihood of the model, i.e., $p(W_2, Z_2, \Theta, \Phi | Z_1, \alpha, \beta)$, where Θ is the topic distribution and Φ is word distributions for topics. To compute the posterior distribution of assigning a topic to a set of word tokens (i.e., $p(Z_2^{W_{2i}} = k | Z_2^{-W_{2i}}, Z_1, \alpha, \beta)$), we integrate Θ and Φ out and compute the posterior as:

$$p(Z_2^{W_{2i}} = k | Z_2^{-W_{2i}}, Z_1, \alpha, \beta) \propto \prod_{d_m \in D_{2i}} (c_{2,m,*}^{k,-W_{2i}} + \alpha)^{c_{2,m,*}^i} \prod_{v \in V_{2i}} \left(\frac{c_{2,*,v}^{k,-W_{2i}} + \beta'_{kv}}{\sum_r c_{2,*,r}^{k,-W_{2i}} + \beta'_{kr}} \right)^{c_{2,*,v}^i}, \quad (3.1)$$

where D_{2i} is the set of documents that contain the word tokens in W_{2i} and V_{2i} is the set of unique words in W_{2i} . The pseudo count β' is the sum of actual counts in cell 1 plus β , i.e., $\beta'_{kv} = c_{1,*,v}^k + \beta$, where $c_{1,*,v}^k$ is the count of assigning topic k to word v in cell 1. The count $c_{2,m,*}^i$ is the number of words in document d_m that are also in W_{2i} . Similarly, $c_{2,*,v}^i$ is the number of occurrences of word v in W_{2i} . The first part of Equation (3.1) is the likelihood of assigning topic k to the current set of documents (D_{2i}) without considering the topic assignments on W_{2i} . The second part of Equation (3.1) is the likelihood of assigning topic k to the unique words in W_{2i} by excluding the topic assignments of W_{2i} . The two parts of Equation (3.1) correspond to the two parts in LDA (Equation (2.1)).

Combining: To combine the topic model of two cells, we apply Equation (3.1) to sample a topic for each word token set. We repeat the sampling process until the complete likelihood converges or the sampling process exceeds a large number of iterations, i.e., 1000. Let $\tilde{c}_{2,*,v}^k$ be the final count of assigning topic k to word v . We combine the topic model by summing up the actual counts of the two cells: $\tilde{c}_{*,v}^k = c_{1,*,v}^k + \tilde{c}_{2,*,v}^k$. Therefore, the word distribution of each topic k for the combined model ϕ_{kv} could be estimated as:

$$\tilde{\phi}_{kv} = \frac{\tilde{c}_{*,v}^k + \beta}{\sum_r \tilde{c}_{*,r}^k + |V|\beta}. \quad (3.2)$$

Complexity: The computation of Equation (3.1) has time complexity $O(|M_{2i}| + |V_{2i}|)$ because we need to compute the production over the documents and words. We have K token sets in total and we need to compute Equation (3.1) for K topics to sample the topic for each token set. Therefore, the total time complexity is $O(K^2(|M_{2i}| + |V_{2i}|)I) = O((K + |M_2| + |V_2|)KI)$, where M_2 is the set of documents and V_2 is the set of unique words in cell 2. Since $K + |M_2| + |V_2|$ is much smaller than $|W_2|$ (e.g., less than 10% of $|W_2|$ for tweets and even lower for long documents).

Sampling Error: We next prove that our algorithm has a bounded error compared to the topic models returned by Online-LDA.

Theorem 1: Let $\phi_k^{(1,2)}$ and $\tilde{\phi}_k^{(1,2)}$ be the language models for topic k after we combine cell 2 to cell 1 using Online-LDA (Equation (2.3)) and our approach (Equation (3.1)), respectively. Let W_1 and W_2 be the word token sets of cell 1 and cell 2, respectively. Then, the expectation of Euclidean distance between two models $\mathbb{E}d(\phi_k^{(1,2)}, \tilde{\phi}_k^{(1,2)}) < \sqrt{2} \frac{|W_1 \cap W_2| + |W_2| - 1}{|W_1| + |W_2|}$. $\square$

Proof sketch: Let $C_2^k = \sum_r c_{2,*,r}^k$, $\tilde{C}_2^k = \sum_r \tilde{c}_{2,*,r}^k$ and $C_1^k = \sum_r c_{1,*,r}^k$. Since the two sampling algorithms are based on the same prior α and $\beta'_{kv} = c_{1,*,v}^k + \beta$, we can assume $C_2^k \approx \tilde{C}_2^k$. This assumption means the two algorithms will results in a similar topic proportion, i.e., the total counts for each topic is similar. Then we infer the bound as:

$$\begin{aligned} \mathbb{E}d(\phi_k^{(1,2)}, \tilde{\phi}_k^{(1,2)})^2 &\approx \mathbb{E} \sum_{r \in W_1 \cap W_2} \left(\frac{c_{2,*,r}^k - \tilde{c}_{2,*,r}^k}{C_2^k + C_1^k + |V|\beta} \right)^2 \\ &\leq \mathbb{E} \sum_{r \in W_1 \cap W_2} \frac{(c_{2,*,r}^k)^2 + (\tilde{c}_{2,*,r}^k)^2}{(C_2^k + C_1^k + |V|\beta)^2} \\ &\leq \frac{\mathbb{E}(\sum_{r \in W_1 \cap W_2} c_{2,*,r}^k)^2 + \mathbb{E}(\sum_{r \in W_1 \cap W_2} \tilde{c}_{2,*,r}^k)^2}{\left(\frac{|W_1| + |W_2|}{|W_2|} C_2^k + |V|\beta \right)^2}. \end{aligned} \quad (3.3)$$

We next apply the definition of variance $\mathbb{E}(X - \bar{X})^2 = \mathbb{E}(X^2) - (\mathbb{E}X)^2$ to the inequality. Let $X = \sum_{r \in W_1 \cap W_2} c_{2,*,r}^k$, then we have:

$$\begin{aligned} \mathbb{E}\left(\sum_{r \in W_1 \cap W_2} c_{2,*,r}^k\right)^2 &= \left(\frac{|W_1 \cap W_2|}{|W_2|} C_2^k\right)^2 \\ &+ \mathbb{E}\left(\sum_{r \in W_1 \cap W_2} c_{2,*,r}^k - \frac{|W_1 \cap W_2|}{|W_2|} C_2^k\right)^2 \\ &< \left(\frac{|W_1 \cap W_2|}{|W_2|} C_2^k\right)^2 + \sum_{r \in W_1 \cap W_2} \mathbb{E}\left(c_{2,*,r}^k - \frac{C_2^k}{|W_2|}\right)^2. \end{aligned} \quad (3.4)$$

Since the maximum value of the second term appears when we assign all words to one topic, i.e., it is not hard to infer that the second term is smaller than $\frac{|W_2| - 1}{|W_2|} (C_2^k)^2$. We apply similar inference on the term $\mathbb{E}(\sum_{r \in W_1 \cap W_2} \tilde{c}_{2,*,r}^k)^2$ in Equation (3.3) to obtain the error bound of combining two topic models: $\sqrt{2} \frac{|W_1 \cap W_2| + |W_2| - 1}{|W_1| + |W_2|}$.

According to Theorem 1, the smaller overlap of words between two cells will yield a lower error bound. Combining small cells to large cells yield a lower error bound.

Combining Multiple Cells: We extend the aforementioned method for combining two Octree cells to multiple cells. Suppose that the given region R and time interval $[t_b, t_e]$

exactly cover n cells, each having a pre-trained topic model. We can apply our combining method $n - 1$ times on the n cells to learn the topic model. Next, we show that the error bound of combining multiple cells is parameterized by the number of combinings.

Theorem 2: Let $\phi_k^{(1,\dots,n)}$ and $\tilde{\phi}_k^{(1,\dots,n)}$ be the language models for topic k obtained by combining topics of n cells according to Equation (2.3) and Equation (3.1), respectively. Let $W_1, \dots, W_n$ be the sets of word tokens of n cells, respectively. Then, the expectation of Euclidean distance between two models $\mathbb{E}d(\phi_k^{(1,\dots,n)}, \tilde{\phi}_k^{(1,\dots,n)}) < f(n) = \sqrt{2} \frac{|\cup_{i=2}^n W_i \cap W_1| + \sum_{i=2}^n |W_i| - 1}{\sum_{i=1}^n |W_i|}$. $\square$

Proof sketch: Let $C_i^k = \sum_r c_{i,*,r}^k$, $\tilde{C}_i^k = \sum_r \tilde{c}_{i,*,r}^k$, where $1 \leq i \leq n$. Similar to Theorem 1, we assume $C_i^k \approx \tilde{C}_i^k$. Then we have:

$$\begin{aligned} \mathbb{E}d(\phi_k^{(1,\dots,n)}, \tilde{\phi}_k^{(1,\dots,n)})^2 &\approx \mathbb{E} \sum_{r \in \cup_{i=2}^n W_i \cap W_1} \left(\frac{\sum_{i=2}^n c_{i,*,r}^k - \tilde{c}_{i,*,r}^k}{\sum_{i=1}^n C_i^k + |V|\beta} \right)^2 \\ &\leq \mathbb{E} \sum_{r \in \cup_{i=2}^n W_i \cap W_1} \frac{(\sum_{i=2}^n c_{i,*,r}^k)^2 + (\sum_{i=2}^n \tilde{c}_{i,*,r}^k)^2}{(\sum_{i=1}^n C_i^k + |V|\beta)^2} \\ &< \frac{2 \left(\frac{|\cup_{i=2}^n W_i \cap W_1| + \sum_{i=2}^n |W_i| - 1}{\sum_{i=2}^n |W_i|} \sum_{i=2}^n C_i^k \right)^2}{\left(\frac{\sum_{i=1}^n |W_i|}{\sum_{i=2}^n |W_i|} \sum_{i=2}^n C_i^k + |V|\beta \right)^2}. \end{aligned} \quad (3.5)$$

Thus, the error bound is $\sqrt{2} \frac{|\cup_{i=2}^n W_i \cap W_1| + \sum_{i=2}^n |W_i| - 1}{\sum_{i=1}^n |W_i|}$.

3.3.2 Processing Partially Covered Cells

Since the given query with spatial boundary R and time interval $[t_b, t_e]$ may not cover an Octree cell completely, but only a part of it, we need to consider how to utilize the partial covered cells. Suppose the query completely covers n cells and partially covers n' cells and let $\tilde{\phi}^{(n)}$ be the language model after combining the n completely covered cells. One solution is to apply Online-LDA to the documents covered by the region and timespan of the given TMRT query in the n' partially covered cells, using $\tilde{\phi}^{(n)}$ as the prior of language models. However, when the number of documents in the n' partially covered cells is large, it is not efficient to apply Online-LDA to these n' partially covered cells.

Notice that for the cell that has a high overlap with R and $[t_b, t_e]$ (e.g., 90% of the documents are covered by R and $[t_b, t_e]$), its pre-trained topic model is often close to the one learned from the covered documents. Therefore, it is desirable to use the pre-trained topic model of the cell to approximate the topic models learned from the covered documents, and apply our proposed fast sampling algorithm presented in Section

3.3.1 to combine it to $\tilde{\phi}^{(n)}$. In this section, we prove that there exists an error bound parameterized by the overlap using the pre-trained topic model to approximate the one learned on covered documents.

Theorem 3: Let ϕ_k^{ALL} and ϕ_k^{PART} be the language models for topic k learned on all the documents and the covered documents of a cell, respectively. Let $p \in (0, 1)$ be the percentage of documents in the cell that are covered by the input spatial boundary and timespan. Then, the Euclidean distance between two models $\mathbb{E}d(\phi_k^{ALL}, \phi_k^{PART}) < \frac{\sqrt{(1+p^2)}}{p}$. $\square$

Proof sketch: Let $c_{*,r}^{k,ALL}$ and $c_{*,r}^{k,PART}$ be the counts of assigning topic k to word r in all the documents and the covered documents, respectively. Let $C^{k,ALL} = \sum_r c_{*,r}^{k,ALL}$, $C^{k,PART} = \sum_r c_{*,r}^{k,PART}$. Without loss of generality, we assume the topics of tokens in the covered documents follow similar distribution of those in all the documents in the cell, i.e., $C^{k,PART} \approx pC^{k,ALL}$. Then, the following inequalities hold:

$$\begin{aligned} \mathbb{E}d(\phi_k^{ALL}, \phi_k^{PART})^2 &\approx \mathbb{E} \sum_r \left(\frac{c_{*,r}^{k,ALL} + \beta}{C^{k,ALL} + |V|\beta} - \frac{c_{*,r}^{k,PART} + \beta}{pC^{k,ALL} + |V|\beta} \right)^2 \\ &< \frac{\mathbb{E}(\sum_r c_{*,r}^{k,ALL})^2 + \mathbb{E}(\sum_r c_{*,r}^{k,PART})^2}{(pC^{k,ALL} + |V|\beta)^2} \end{aligned} \quad (3.6)$$

Because $C^{k,PART} \approx pC^{k,ALL}$, the numerator in Equation (3.6) is $(1 + p^2)C^{k,PART}$. Thus, we have $\mathbb{E}d(\phi_k^{ALL}, \phi_k^{PART}) < \frac{\sqrt{(1+p^2)}}{p}$.

According to Theorem 3, we combine the topic models of partially covered (at least τ_p percent of coverage) cells to those of completely covered cells with bounded error. Then for the remaining documents that are covered by the TMRT query, but are not in the cells covered by the query completely or at least τ_p percent, we use Online-LDA to sample the topics for them.

3.3.3 Octree based Pre-Computation

After presenting our techniques for combining pre-computed topic models, we are now ready to present our techniques for the offline pre-computation process. The topic models built in the offline phase will be used to support our online topic mining algorithm.

In the offline pre-computation process, we need to consider two important research problems. First, how to divide the spatio-temporal documents in a large Octree cell into smaller cells to answer TMRT query efficiently and effectively, namely **Octree Cell Division** problem, which affects the accuracy and efficiency of our combining algorithm, and thus the efficiency and effectiveness of answering TMRT queries. Second, how to

decide on which Octree cells we pre-train a topic model to balance the efficiency, storage, and accuracy of our approach, namely **Octree Cell Pre-Training** problem. Next, we present our approaches to solving the two problems.

Octree Cell Division: One straightforward way to divide a large Octree cell into smaller ones is to divide it equally in both spatial and temporal dimensions. However, this method is not optimized for our combining algorithm because it does not consider the word overlap between the child cells, which affects the accuracy of our algorithm for combining topic models as given in Theorem 1.

We propose to select the division point for each dimension (latitude, longitude and time) by minimizing the word overlap between the 8 child cells divided by the division point. Specifically, suppose D_p be the set of documents in the Octree cell to be divided, $D_p^{(i)}(x)$ be the i -th child cell divided by 3-dimensional point x (latitude, longitude, and time). We optimize the following objective function:

$$\begin{aligned}
 & \arg \min_x \sum_{i=1}^8 \sum_{j=1, j \neq i}^8 |\{\cup_{d \in D_p^{(i)}(x)} \{d.w\}\} \cap \{\cup_{d \in D_p^{(j)}(x)} \{d.w\}\}| \\
 & s.t. \quad \frac{|\{d | d^{(h)} > x^{(h)}, d \in D_p\}|}{|D_p|} > \sigma, \\
 & \quad \frac{|\{d | d^{(h)} \leq x^{(h)}, d \in D_p\}|}{|D_p|} > \sigma, \\
 & \quad \forall h \in \{latitude, longitude, time\},
 \end{aligned} \tag{3.7}$$

where $d^{(h)}$ is one of the three dimensions of document d , $x^{(h)}$ is one of the three dimensions of the division point x , and σ is the threshold to control at least how many percentages of documents should be assigned to child cells on both sides of each dimension after division. This treatment is important to balance the size and overlap of the two sub-cubes. An extreme case could happen when we divide the cell into 7 empty cells and a cell that contains all the documents to achieve zero overlap. However, this division is meaningless for accelerating online topic learning. Our treatment can avoid the extreme case. More importantly, our method for Octree cell division can achieve a smaller error bound than the straightforward method of dividing a cell into eight equal-sized cells according to Theorem 1. When the division point of the straightforward method does not satisfy Equation (3.7), the overlap between any two sub-cubes is not minimized, and thus the error bound of the straightforward method is larger than that is obtained by the optimal division point in Equation (3.7) according to Theorem 1.

We solve this optimization problem as follows. Documents that fall in the cell to be partitioned comprise the set of candidate division points. We traverse these documents

and select the point (document) that achieves the minimum objective score defined in Equation (3.7). For each candidate point, it takes $O(|W'_p|)$ to compute Equation (3.7), where W'_p is the set of word tokens in the documents within the constraints D'_p . Totally we have $|D'_p|$ candidate points, and thus the worst case time complexity is $O(|D'_p||W'_p|)$.

Octree Cell Pre-training: There exist two trade-offs in selecting the cells for topic model pre-computation. First, learning topics for cells in higher levels help improve accuracy of our method by reducing the number of combining operations if the region and timespan in the query cover the high-level cell. Nevertheless, a cell from higher levels is less likely to be completely covered by a user specified region and timespan. TMRT queries with small regions and narrow timespans may not contain the high level cells, and thus we cannot use the pre-trained topic models on high level cells to efficiently answer those TMRT queries. Second, if we build a deep Octree and pre-trained topic models for all cells, there is a better chance we can leverage the pre-trained models by our combining algorithm for a TMRT query. However, learning topic models for all cells in a deep Octree is space and time consuming.

Specifically, to make our online topic combining algorithm efficient and effective, we want to choose a group of Octree cells such that 1) the topic models built on these cells do not exceed a given space threshold τ_s ; and 2) it is guaranteed that the error bound of combining topic models in the nearest descendants of these cells does not exceed a given threshold τ_e . This is because even though the TMRT query does not completely cover a high level cell, we can still combine the lower level cells covered by the query without exceeding the error threshold τ_e . To achieve this goal, we propose a greedy division algorithm for building the Octree and select the cells for pre-computation as described in Algorithm 1.

The main idea of the algorithm is to divide the Octree as finer-grained as possible within the space and accuracy threshold. We first insert all the spatio-temporal documents to the root of the Octree (line 1). To efficiently rank and find the largest cell, we maintain a max heap for cells to be divided with the number of spatio-temporal documents contained in each cell as the key (line 2). Each time, we try to divide the largest cell (that contains largest number of spatio-temporal documents) into 8 subregions based on Equation (3.7) (lines 5 – 6). We invoke function `UpdateModelPos` (line 7 and line 10) to estimate the number of topic models we need to learn to guarantee the accuracy threshold τ_e . If the number of models exceeds the space threshold τ_s , we withdraw the division of current cell and stop dividing the Octree (lines 8 – 11); Otherwise, we insert the children of the current cell to the max heap to further divide them (lines 12 – 14). We repeat this process until the number of models exceeds the space threshold.

The function `UpdateModelPos` updates the number of models recursively from bottom to top. We aggregate the number of models to be learned in the descendant cells of the current cell (lines 4 – 5). Suppose that the number of models in the descendants is

subCount. Then we need $subCount - 1$ combining operations to construct a topic model for the current cell. We can measure the error bound of the $subCount - 1$ combining operations using a function of the number of combining operations, i.e., $f(subCount - 1)$. When the error bound is larger than the accuracy threshold τ_e , we train a topic model for the current cell to ensure that the error does not exceed the accuracy threshold (lines 6 – 8). We introduce an attribute *placeModel* to mark the cells we need to learn a topic model.

After applying Algorithm 1, we traverse the Octree and check the *placeModel* attribute of each cell. If it is true, we learn a topic model for the cell.

3.3.4 Topic Mining within Region and Time Interval

Given an Octree with pre-trained topic models, we are now ready to present our online topic mining process for a TMRT query with region R and interval $[t_b, t_e]$ in Algorithm 2. We first find the cells that are covered by the given region at least τ_p percent and contain pre-trained topic models (line 1, **CellSelection**). Then, we order the returning cells in the descending order of their size (line 2) because combining smaller cells to larger cells could be more efficient as discussed in Section 3.3.1. Next, we combine the models of those cells using our fast word set based sampling algorithm (line 5, **FastSetSampling**). Finally, we invoke Online-LDA to handle the remaining documents that are not covered by the founded cells.

In Algorithm 2, the function **CellSelection** returns cells that are completely or highly covered by the given TMRT query $Q = \{R, [t_b, t_e]\}$. When we select cells to combine, we aim to reduce the number of combining operations for 1) reducing the total combining time cost, and 2) reducing the error of combining. Therefore, we follow the greedy principle to chooses cells as large as possible (Algorithm 3). The sets $T.D$ and $Q.D$ are the sets of documents covered by the current cell T and query Q , respectively.

In Algorithm 3, we traverse the Octree tree in a breadth-first manner. We add the current cell to the result set if the cell is completely or highly covered by the given query Q (line 2). We continue to search lower levels if the current cell has overlap with Q (lines 4 – 6). We use a predefined coverage threshold τ_p to decide whether a cell is highly covered by Q and the error bound has been proven in Theorem 3.

3.4 Experiments

We introduce the experimental setting in Section 3.4.1. We report the offline pre-computation cost of our framework in Section 3.4.2, and the experimental results in Sections 3.4.3 and 3.4.4. We show some example topics discovered by our algorithm in Section 3.4.5.

Algorithm 1: GreedyDivision**Input:** A set of spatio-temporal documents D , space threshold τ_s , error threshold τ_e **Output:** Qctree T **begin**

```

1  Initialize  $T$  and Insert  $D$  to  $T$ ;
2  Initialize max heap  $H$ ,  $H.Insert(T)$ ;
3   $modelCount \leftarrow 0$ ;
4  while  $modelCount < \tau_s$  do
5       $T' \leftarrow H.DeleteTop()$ ;
6       $T'.Subdivide()$ ;
7       $modelCount \leftarrow UpdateModelPos(T, \tau_e)$ ;
8      if  $modelCount > \tau_s$  then
9          Withdraw the subdivision for  $T'$ ;
10          $modelCount \leftarrow UpdateModelPos(T, \tau_e)$ ;
11         return  $T$ 
12     foreach  $T_c$  in  $T'.Children$  do
13         if  $T_c.D \cap D \neq \emptyset$  then
14              $H.Insert(T_c)$ ;
15 return  $T$ 

```

Function $UpdateModelPos(T, \tau_e)$

```

1   $modelCount \leftarrow 0$ ;
2  if  $T$  is not leaf then
3       $subCount \leftarrow 0$ ;
4      foreach  $T_c$  in  $T.Children$  do
5           $modelCount \leftarrow subCount + UpdateModelPos(T_c, \tau_e)$ ;
6      if  $f(subCount - 1) > \tau_e$  then
7           $modelCount \leftarrow modelCount + 1$ ;
8           $T.placeModel \leftarrow True$ ;
9      else
10          $modelCount \leftarrow modelCount + subCount$   $T.placeModel \leftarrow False$ ;
12 else if  $T.D \cap D \neq \emptyset$  then
13      $modelCount \leftarrow 1$ ;
14      $T.placeModel \leftarrow True$ ;
15 return  $modelCount$ ;

```

3.4.1 Experimental Setup

Data Sets Our experiments are conducted on two real-life datasets collected from Twitter and Meetup, respectively. We use the Meetup dataset to evaluate the performance of our solution in longer text. Dataset Twitter comprises 7 million tweets posted from

Algorithm 2: OnlineLearning

Input: Octree T , spatio-temporal query $Q = \{R, [t_b, t_e]\}$, coverage threshold $\tau_p \in (0, 1]$
Output: Topic model within Q
begin
 1 $N \leftarrow \text{CellSelection}(T, Q, \tau_p)$;
 2 Sort N by descending order of cell size;
 3 $Model \leftarrow N[0].Model$;
 4 **for** $i = 1$ **to** $N.length$ **do**
 5 $Model \leftarrow \text{FastSetSampling}(Model, N[i].Model)$;
 6 $D_{remain} \leftarrow \{d | d \notin N, d \in Q\}$;
 7 $Model \leftarrow \text{OnlineLDA}(Model, D_{remain})$;
 8 **return** $Model$

Algorithm 3: CellSelection

Input: a Octree T , spatio-temporal query $Q = \{R, [t_b, t_e]\}$, coverage threshold $\tau_p \in (0, 1]$
Output: cell set N
begin
 1 $N \leftarrow \emptyset$;
 2 **if** $(\frac{|T.D \cap Q.D|}{|T.D|} > \tau_p)$ **and** T has pre-trained topic model **then**
 3 $N \leftarrow N \cup \{T\}$;
 4 **else if** $T.D \cap Q.D \neq \emptyset$ **then**
 5 **foreach** T_c in $T.Children$ **do**
 6 $N \leftarrow N \cup \text{CellSelection}(T_c)$;
return N ;

Aug 2012 to Oct 2012 in United States. Each tweet has text, location (latitude and longitude), and creation time, and each tweet contains 9 words on average. Dataset Meetup comprises 1.1 million events posted from Jan 2006 to Jun 2015. Each event has text description, location and creation time. and each Meetup event description has around 109 words on average. For Twitter, we randomly hold out 20% tweets of each day as test data to evaluate the accuracy of the topic models. For Meetup, we randomly hold out 20% words of each document as documents in test data to evaluate the accuracy of the topic models. The details of the two datasets are listed in Table 3.4.

Baseline Methods We compare our method using fast set sampling (FFS) with the following four methods.

LDA: We learn topics for the documents in the given region and timespan using LDA [BNJ03] for each TMRT query.

Online-LDA: We leverage the Octree structure in our framework to find the largest

Table 3.4: Dataset statistics

Datasets	Twitter	Meetup
# Documents	7,012,778	1,099,614
# Word Tokens	69,075,864	120,484,706
Vocabulary Size	137,141	88,004
Time Duration	Aug 2012 - Oct 2012	Jan 2006 - Jun 2015

cell that is covered by the given region and timespan, and apply Online-LDA [ABD08] to sample the remaining documents based on the topic model of the largest cell.

FFS-EQ: We use our fast sampling algorithm to learn topics for TMRT queries. The difference from FFS is that FFS-EQ partitions a cell into 8 qual-sized smaller cells when building the Octree.

 Figure 3.3: Offline pre-computation cost w.r.t. accuracy threshold τ_e

Evaluation Metrics To evaluate the efficiency and accuracy of the two models, we randomly generate 100 pairs of region and timespan with random size, each containing at least 100 documents in the test set for accuracy evaluation. For efficiency, we report the average runtime of the 100 pairs of region and timespan for all methods. To evaluate accuracy, we use perplexity to measure how likely we can use the model to generate the test set, which is a commonly used measure to evaluate the quality of topic models [WMSM09]. Lower perplexity means that the model has better ability of generalization. Perplexity is computed by using the probabilistic model learned from the training set to estimate the likelihood of generating the test set. Specifically, it is computed as follows:

$$\text{perplexity}(W_{test}) = \exp\left\{-\frac{\log p(W_{test}|\Phi_{train})}{|W_{test}|}\right\}, \quad (3.8)$$

where $|W_{test}|$ is the set of word tokens in the test set, and Φ_{train} is the topic model to be evaluated. We apply the “left-to-right” method [WMSM09], which is often used

 Figure 3.4: Offline pre-computation cost w.r.t. space threshold τ_s

 Figure 3.5: Offline pre-computation cost w.r.t. number of topics K

for evaluating topic models, to estimate perplexity. We report the sum of perplexity of models learned from the 100 pairs of region and timespan.

Parameter Settings We have three parameters in our framework, i.e., accuracy threshold τ_e , space threshold τ_s , and number of topics K . Note that the accuracy threshold τ_e is in fact a function of number of cells (n) to combine $f(n)$. For simplicity, we use $\tau_e = f(n)$ to denote the accuracy threshold of combining n cells. The space threshold specifies the maximum number of topic models to pre-train. We analyze the effect of these parameters by varying one parameter while fixing the others as default values. The settings of each parameter is shown in Table 3.5. The default settings of these parameters are $\tau_e = f(8^2)$ (i.e., learning topic models every 2 levels in the tree), $\tau_s = 5000$ and $K = 100$. We set the constraint for cell division σ in Equation (3.7) at 0.3. Online-LDA and FFS-EQ use the same settings as our method (FFS).

All algorithms are implemented in C# on a workstation with Intel(R) Xeon(R) CPU E5-2680 v2 @2.80GHz and 64GB RAM.

Table 3.5: Parameter settings

Parameter (notation)	Settings	Default
accuracy threshold (τ_e)	$f(8^1)$ to $f(8^5)$	$f(8^2)$
space threshold (τ_s)	5K to 20K (topic models)	5K
number of topics (K)	20 to 100	100

 Figure 3.8: Runtime w.r.t. space threshold τ_s

 Figure 3.9: Perplexity w.r.t. space threshold τ_s

3.4.2 Pre-computation Cost

We report the pre-computation cost of FFS-EQ and our method FFS with different parameter settings. From Figure 3.3, we observe that the pre-computation cost is higher when the accuracy threshold τ_e is set at smaller value (corresponding to smaller error). This is because we need to learn topic models in higher level cells (which contain more documents) in the Octree to achieve better accuracy. Similarly, as the space threshold τ_s or the number of topics increases, the pre-computation costs of the three methods increase as shown in Figure 3.4 and Figure 3.5.

Our framework (FFS) with default setting takes around 100 minutes for pre-computation on Twitter data and 300 minutes on Meetup data. The pre-computation time of FFS and FFS-EQ is similar because the computation cost of pre-training topic models overwhelms the computation cost of our cell division method as described in Equation (3.7).

 Figure 3.10: Runtime w.r.t. number of topics K

 Figure 3.11: Perplexity w.r.t. number of topics K

3.4.3 Evaluation on Different Parameters

Varying Accuracy Threshold τ_e The accuracy threshold is used to guarantee the error bound of topic combining when we build the Octree as discussed in Section 3.3.3. We set the threshold as $f(8)$, $f(8^2)$, $f(8^3)$, $f(8^4)$, and $f(8^5)$, which indicates that we pre-train topic models every 1, 2, 3, 4, and 5 levels, respectively. We report the running time of learning topic models for the 100 random pairs of region and timespan in Figure 3.6(a) and Figure 3.6(b) as we vary the accuracy threshold.

Our proposed framework FFS is more than an order of magnitude faster than LDA and Online-LDA under different accuracy thresholds because our proposed sampling algorithm can efficiently combine two models without sampling all the word tokens. We also observe that with a smaller accuracy threshold (smaller τ_e), the online topic learning is slightly faster. With a lower accuracy threshold, we learn more pre-trained models for large cells

Figure 3.12: Runtime w.r.t. region size

Figure 3.13: Perplexity w.r.t. region size

in an Octree. Larger cells are more likely to cover most documents in the specified region and timespan, and thus reduce the time for combining topic models.

Figure 3.7(a) and Figure 3.7(b) show the perplexity computed on the test data for all methods. The perplexity of our proposed framework FFS is close to those of LDA and Online-LDA because our algorithm combines topic models with a bounded error. Benefiting from the proposed partitioning method of Octree cells, FFS achieves higher accuracy in most cases (lower perplexity) compared with FFS-EQ.

Varying Space Threshold τ_s With a larger space threshold, we have more space for pre-trained topic models, and thus we can construct a deeper Octree. A large space threshold may help accelerate the online learning process because we can learn more pre-trained models in the Octree. As shown in Figure 3.8(a) and Figure 3.8(b), FFS and FFS-EQ outperform LDA and Online-LDA by more than an order of magnitude. As shown

Figure 3.14: Runtime w.r.t. timespan size

Figure 3.15: Perplexity w.r.t. timespan size

in Figure 3.9(a) and Figure 3.9(b), FFS has better accuracy than FFS-EQ in most cases.

Varying Number of topics K The number of topics K affects both Octree based pre-computation and the running time of all methods because they all have complexity linear to K . As shown in Figure 3.10(a) and Figure 3.10(b), all methods run slower as K increases. The running time of our framework FFS increases a bit faster than LDA and Online-LDA because the complexity of our framework is quadratic to K . However, in practice K is often set to small values. As shown in Figure 3.11(a) and Figure 3.11(b), the perplexity increases when the number of K increases because there are more parameters for topic models (ϕ_k). When the number of parameters increases, the likelihood computed as the joint probability of word tokens decreases.

3.4.4 Evaluation on Different Regions and Timespans

We also experiment on different user specified regions and timespans. All the parameters are set as default values except the space threshold τ_s is set at 20000. Let the spatial area of the document collection be A . We vary the size of regions with the following values: $0.0001A$, $0.001A$, $0.01A$, $0.1A$ while fixing the size of timespans as 5 days on Twitter data and 300 days on Meetup data. We randomly generate 100 pairs of region and timespan for each of the four settings of region size. The efficiency results are shown in Figure 3.12(a) and Figure 3.12(b), and the perplexity results are shown in Figure 3.13(a) and Figure 3.13(b). As region size decreases, the running time decreases while the perplexity increases. The running time of our algorithm is close to or even slightly higher than LDA and Online-LDA because when the region becomes smaller than the leaf cell of the Octree, our framework is equivalent to LDA. The perplexity of all methods decreases because the number of documents covered by the region increases. Without enough documents (When region size is $0.0001A$), each region contains only several hundred or several thousand documents on average for both dataset, the learned LDA model is less effective on the hold-out test data.

We also vary the timespan by 5, 10, 15, 20 days on Twitter Data and 300, 600, 900, 1200 days on Meetup data while fixing the size of regions to $0.1A$. For each of the timespan, we randomly generate 100 pairs of region and timespan. Figure 3.14(a) and Figure 3.14(b) show the efficiency of all methods in different timespans, while Figure 3.15(a) and Figure 3.15(b) show the perplexity. With a larger timespan, more documents are included, and thus it takes longer to process. The perplexity of FFS and FFS-EQ is similar to those of LDA and Online-LDA as shown in Figure 3.15(a).

3.4.5 Qualitative Evaluation on Extracted Topics

To exemplify the extracted topics of the three methods, we submit a TMRT problem within the region of New York city and timespan from Sep 1, 2012 to Sep 30, 2012, as shown in Figure 3.16. We apply FFS, LDA and Online-LDA to this TMRT problem using the default parameter setting. The number of words is 637,581, we run 1000 iterations of Gibbs sampling for the three methods. The running time of FFS, LDA and Online-LDA are 104.96 seconds, 1586.14 seconds and 1776.38 seconds, respectively.

We randomly select three topics from LDA and find three topics from the result of Online-LDA and our proposed framework (FFS) that are most similar to (measured by KL divergence) the three ones of LDA, respectively. We manually name the extracted three topics as “Photo Sharing”, “Politics” and “Celebration”, and visualize them using “Word Cloud” in Figure 3.16.

In general, the topics extracted by the three methods are similar. Surprisingly, our proposed framework can also discover some representative words for a topic (e.g., “friend”

Figure 3.16: Topics extracted by different methods

in topic “Photo Sharing” and “work”, “education” in topic “Politics”) that LDA and Online-LDA have not discovered. Consider the quality of extracted topics and runtime efficiency, our proposed framework is more suitable to solve the TMRT problem compared to LDA and Online-LDA.

3.5 Summary

We study the problem of mining topics from a collection of spatio-temporal documents given a region and a time interval. Based on the LDA model, we propose an algorithm

with bounded errors to combine two pre-trained topic models learned from two document sets, each falling in a region and a time period. We also propose a method to build an Octree based indexing structure to facilitate our online topic learning algorithm. We partition the Octree cells by considering the word overlap of the generated cells to achieve better accuracy for combining two pre-trained topic models. Then we employ our proposed algorithms to combine the pre-trained topic models to obtain the topics for documents in the specified region and time period. Our experimental results suggest that our proposed framework is able to improve the runtime performance by at least an order of magnitude compared with LDA and Online-LDA. Meanwhile, the effectiveness of our proposal, measured by perplexity, is close to that of LDA and Online-LDA.

This work combines data management principles to improve the efficiency of machine learning/data mining algorithms. This work opens to a number of promising directions for future work. For example, exploratory topic mining from spatio-temporal streams, topic trends detection in a user specified region and timespan, and extended TMRT query with keywords as another query dimension.

Chapter 4

Distributed Training of Geographical Topic Models

The effectiveness of applying geographical topic models to uncover topics in latent regions have been proven in many recent studies [Siz10, YCH⁺11, LFYX13, YCM⁺13, AHS13, LXP⁺15, LEHC15]. These studies mainly focus on the effectiveness of the model while ignoring the training efficiency. In practice, training geographical topic models is too time-consuming to be applied on large datasets. It often takes months to train a model with thousands of topics and thousands of regions on millions of documents, and there exists no distributed solution for training geographical topic models. Although there exist several solutions [NASW09, AAG⁺12, LZCS11, ZR13, WZS⁺14, YGH⁺15, YHY⁺15, CLZC16] tailored for LDA, they are not applicable to our problem because geographical topic models often have a topic-region matrix which is difficult to synchronize efficiently. In this chapter, we propose a distributed solution for training geographical topic models on large datasets. Our solution takes account of three objectives — (1) support parallelism of computation and network communication; (2) reduce memory requirement; and (3) reduce network communication cost. Experiments show that the proposed solution is able to train large models with millions of parameters from tens of gigabytes of geo-textual documents within 20 hours on a small cluster of 20 machines.

The remaining part of this chapter is organized as follows. Section 4.1 gives a formal definition of the problem. Section 4.2 starts with a baseline solution and then represents the proposed solution which achieves three objectives: support parallelism, reduce memory requirement, and reduce communication cost. Section 4.3 provides the implementation details and Section 4.4 evaluates the proposed solution on real world datasets. Section 4.5 summarizes the key findings of this chapter.

4.1 Problem Definition

4.1.1 Data and Model Formulation

Definition 3: Geo-textual Document. A geo-textual document is a tuple $d = \langle \mathbf{w}_d, \mathbf{l}_d \rangle$, where $\mathbf{w}_d$ is the set of word tokens in the documents and $\mathbf{l}_d$ is the latitude-longitude geo-location of the document. $\square$

Example geo-textual documents include geo-tagged tweets, check-ins in location based services such as Foursquare, geo-tagged news, geo-tagged web pages, etc.

Definition 4: Geographical Topic Model. Given a set of geo-textual documents D , the number of topics K , and the number of regions R , geographical topic models are to discover K latent topics, R latent regions, and the topic distribution of regions $\psi \propto m_{kr} + \gamma_{kr}$. Each topic z is a word distribution $\phi_z \propto c_{*,w}^k + \beta_{zw}$ and each region r is a continuous Gaussian distribution $\mathcal{N}(\mu_r, \Sigma_r)$. $\square$

4.1.2 Distributed Training of Geographical Topic Model

Consider that the data is often much larger than the model, we adopt the parameter server setting [HCC⁺13, LAP⁺14, SN10] as does the recent study on distributed LDA [YGH⁺15, CLZC16]. That is, we transfer the model instead of the data via network. In the parameter server architecture, the memory of each machine is divided by two parts - (1) worker memory and (2) the shared memory which is regarded as the “server memory” of the parameter servers. The model parameters are distributed and stored in the shared memory of each machine, i.e., the parameter server. Under this setting, the problem is formulated as follows:

Definition 5: Distributed Training of Geographical Topic Models. Given a set of geo-textual documents D , and a cluster of M machines, the task of distributed training of a geographical topic model is to distribute D to the M machines and learn the model, i.e., K topics, R regions and the correlation of topics and regions, on the data distributed to the workers in parallel. $\square$

4.2 Distributed Training for Geographical Topic Models

We first present a straightforward solution for distributed training of geographical topic models and identify issues with the straightforward solution. To address these issues, we propose three design objectives (Section 4.2.1). Then, we present the proposed training

algorithm and model parallelism (Section 4.2.2) to achieve highly parallel training and reduce the local memory requirements for each machine. We present a parameter localization method to reduce the communication cost (Section 4.2.3). We also propose a new training algorithm that is able to reduce the training complexity significantly with a bounded error in Section 4.2.4. Table 4.1 lists the notations used in this chapter.

Table 4.1: Summary of notations

Notation	Description
K, R	Number of topics and number of regions
$z^{(m,n)}$	The topic assignment for the n -th word token in document m
r_m	The region assignment for document m
ψ_r	The topic distribution of region r
ϕ_k	The word distribution of topic k
μ_r, Σ_r	The mean and variance of region r
m_{kr}	Number of co-occurrences of the topic k and r in the same document
$c_{*,w}^k$	Frequency of assigning topic k to word w
c^k	Frequency of topic k in the dataset
c_r	Frequency of region r in the dataset
α, β, γ	Hyper-parameters of the model

4.2.1 Baseline Solution and Overview of the Proposed Solution

We first present a straightforward solution for distributed training of geographical topic models, and identify its major issues. To address these issues, we propose three design objectives. Finally, we will present an overview of the proposed solution, aiming to achieve the three objectives.

Straightforward Solution. A distributed solution to training a geographical topic model is to evenly distribute the data randomly to each worker. The global parameters (e.g., the topic-region and word-topic matrices, the region parameters μ_r, Σ_r , and the topic counts $c^k = \sum_v c_{*,v}^k$) are stored in the parameter servers, i.e., the shared memory across the machines. Moreover, each worker machine stores a local copy of the parameters. When parameters cannot be fit into the memory of workers, paging algorithm, e.g., Least Recently Used (LRU) is used to swap the parameters between memory and hard disk. In each iteration, each worker needs to fetch the global parameters, then Equation (2.4) is applied to sample the topic and region for each word token in the documents assigned to the worker, and finally its local copy of parameters is updated. At the end of each iteration, the synchronization of the updates of the parameters among machines is scheduled.

The shortcomings of this solution are three-fold. Firstly, the computation and network communication cannot be conducted in parallel. The training algorithm cannot run until the updated parameters are fetched from the parameter servers. Secondly, the model often cannot fit into memory. For example, a model with 100K topics and regions trained on a dataset with a moderate size of vocabulary (300K) takes up to 160 GB memory space. The solution has to frequently swap the parameters between memory and hard disk, incurring expensive I/O cost. Thirdly, access and synchronization of the parameters during each training iteration incurs heavy network communication cost.

Design Objectives. To address these issues, we propose the following objectives for designing a distributed solution to train a geographical topic model.

1. Support the parallelism between computation and network communication - The worker machines can start training before all parameters are loaded.
2. Reduce memory requirement - Ideally, each time a small subset of the parameters are fetched via network and held in memory for learning the topics and regions.
3. Reduce communication cost - Instead of accessing all of the parameters in each iteration, we aim to achieve parameter localization such that each worker needs to access only part of the parameter and the worker machines only fetch the parameters via network when necessary.

To achieve the three objectives, we need to address the following challenges. First, to support the parallelism between computation and network communication while reducing memory requirement in each worker machine, a natural idea is as follows: We partition the parameters into distinct subsets; Each time the worker machine samples the topic and region for each word token based on a subset of parameters, it pre-fetches the next subset of parameters from the parameter servers at the same time. However, such model parallelism is difficult to achieve — the Gibbs sampling algorithm requires both the word-topic matrix and the topic-region matrix to learn the topic and region for each word token, and thus all the parameters have to be fetched before sampling. The requirement makes it difficult to parallelize the computation and network communication, and also incur expensive requirement on memory. Second, to reduce the communication cost of each worker machine, ideally we want to achieve parameter localization and fetch parameters in an on-demand manner. However, both region and topic for each word token are unknown before training. It is difficult to achieve parameter localization and on-demand fetching parameters.

PGeoTopic. Based on the design objectives, we propose a distributed solution for training geographical topics, called *PGeoTopic*. We tackle the aforementioned challenges as follows: First, to support parallelism between computation and network communication

Figure 4.1: Training effectiveness and efficiency (in 100 iterations) of different proposals on a random sample of 10K geo-tagged documents from Twitter dataset. The number of topics and regions are set at 10. **Uniform** is the uniform proposal, and **PGeoTopic** is our method. The numbers N in **PGeoTopic- N** and **Uniform- N** are metropolis steps.

while reducing the memory requirement, we propose a new training algorithm (Section 4.2.2.1) and a model parallelism (Section 4.2.2.2). The proposed training algorithm decouples the interdependency of the word-topic matrix and the topic-region matrix, and then supports using either word-topic matrix or topic-region matrix to sample the topics and regions for word tokens. The isolation of the two matrices allows fetching the topic-region matrix while sampling using the word-topic matrix and vice versa. The model parallelism partitions the word-topic matrix by words and the topic-region matrix by regions. We name each partition of the word-topic matrix and the topic-region matrix as *word slice* and *region slice*, respectively. The model parallelism enables each worker machine to fetch and store only one word/region slice in the local memory each time, greatly reducing the memory requirement. Second, we propose to adopt spatial partitioning to make data allocated to each worker is spatially close (Section 4.2.3.1), and then achieve parameter localization to reduce communication cost in two aspects — (1) we propose a two-step sampling scheme (Section 4.2.3.2) to support on-demand fetching only parameters of relevant regions for each worker; (2) We propose to assign the global parameters that are related to a worker to the shared memory of the worker. Accessing parameter servers only needs access local machine, thus reducing communication cost (Section 4.2.2.2). Third, we further develop a new algorithm on top of the proposed training algorithm to reduce the sampling complexity for each word token to $O(1)$ with bounded errors (Section 4.2.4).

4.2.2 The Proposed Training Algorithm & Model Parallelism

We parallelize the computation and network communication while reducing the memory requirement in two aspects — (1) We decouple the word-topic matrix and the topic-region matrix to allow independent access to the two matrices for training geographical topic models (Section 4.2.2.1); (2) We propose a model parallelism that further improves the granularity of the parallelism and reduces the memory requirement in each worker machine (Section 4.2.2.2).

4.2.2.1 Decoupling the Parameter Matrices

Consider sampling the topic and region for a word token using Equation (2.4). Even though we can only fetch the topic vector of the current word token from the word-topic matrix, the whole topic-region matrix is needed before sampling a topic and region for any word token. This interdependency between the two parameter matrices incurs extremely high network communication before the sampling computation. To solve the problem, we propose to decouple the dependency of the two matrices and make use of them independently. The high-level idea is that we divide the word tokens into two sets, and for one set we use the word-topic matrix and the region parameters (e.g., $f(l_d|\mu_r, \Sigma_r) \cdot (c_r + \alpha_r)$ in Equation (2.4)) to sample topics and regions while using the topic-region matrix to sample topics and regions for the other set. Decoupling the two matrices enables the parallelism between computation and network communication in that the worker machines can fetch the topic-region matrix from parameter servers while they are using the word-topic matrix for training. Furthermore, finer-grained parallelism can be achieved by partitioning the two matrices into slices after decoupling the two matrices, which will be presented in Section 4.2.2.2.

To realize the high-level idea, we propose a new training algorithm that uses Metropolis-Hastings [Has70] as building block to decouple the two matrices.

Metropolis-Hastings (MH) sampling algorithm is often used to draw a sample from distributions that are hard to sample from. The idea behind this algorithm is to use another distribution, which is called “proposal”, to draw a sequence of samples. For each sample, the algorithm judges whether the sample should be accepted or rejected. The current sample is accepted when it is more likely to be sampled from the original distribution than the previous one. More precisely, current sample x_t is accepted with the probability $\min(1, \frac{p(x=x_t)q(x=x_{t-1})}{p(x=x_{t-1})q(x=x_t)})$, where x_{t-1} is the previous sample, $p(x)$ and $q(x)$ are the density (at x) of the original distribution and the proposal distribution, respectively. The number of samples required to mix (i.e., samples become close to the original one) is often called Metropolis steps or burn-in steps. The samples after burn-in steps are used to approximate samples drawn from the original distribution. When the proposal is

similar to the original distribution, only a few Metropolis steps are needed. Otherwise, Metropolis-Hastings takes long time to simulate the original distribution.

In our case, the original distribution is the posterior in Equation (2.4), and we aim to derive a proposal distribution that supports independent access to the two global matrices.

Challenges. The Metropolis-Hastings algorithm opens opportunities to change the way of sampling. However, constructing a proper proposal distribution is usually difficult. The proposal distribution should be able to draw similar samples as does Equation (2.4), and be able to converge to similar likelihood within similar number of iterations as does Equation (2.4). Meanwhile, the sampling efficiency should be as high as possible, and supports independent access to the two matrices. Figure 4.1 shows the training effectiveness and efficiency of some example proposals. Uniform distribution with 10 Metropolis steps (**Uniform-10**) is the most efficient. It does not need to access the matrices but fails to improve the log-likelihood. To the other extreme, directly sampling from the posterior (**True posterior**) needs to access both matrices and is often time-consuming. Our goal is to construct a proposal like **PGeoTopic** in Figure 4.1 that achieves effective and efficient training and supports independent access to the word-topic and topic-region matrices.

Observations. We develop our training algorithm based on several observations to tackle the aforementioned challenges. Recall that the posterior distribution in Equation (2.4) has three terms corresponding to word-topic relation, topic-region relation and location-region relation, respectively.

- **Observation 1:** The first and last terms are only related to topics and regions, respectively. Sampling a topic from the word-topic distribution and a region from the posterior of Gaussian mixture can be viewed as fitting a topic model and a Gaussian mixture model independently, i.e., $p(z^{(m,n)} = k, r^{(m,n)} = r | D) = p(z^{(m,n)} = k | D) \cdot (r^{(m,n)} = r | D)$.
- **Observation 2:** The topic-region correlation term m_{kr} captures the interdependency between topic and region. Sampling from this term is similar to sampling from a data-independent distribution of topic and region, i.e., $p(z^{(m,n)} = k, r^{(m,n)} = r)$.
- **Observation 3:** The final proposal should consider all the three terms in Equation (2.4) to draw similar samples as does Equation (2.4).

Motivated by the first two observations, we propose two proposal distributions, namely independent-proposal and joint-proposal, each of which preserves parts of the statistics (i.e., the topic-region correlation, word-topic relations and the region distribution) in

Equation (2.4). The independent-proposal samples the topics according to the word distribution, and the region according to the Gaussian mixture model, i.e., the last term in Equation (2.4). The joint-proposal samples the topic-region pair according to the topic-region correlation.

Independent-proposal. We define the independent-proposal as:

$$q_{indep}(z^{(m,n)} = k, r^{(m,n)} = r) \propto \frac{c_{*,w}^k + \beta_{kw}}{c^k + \bar{\beta}} \cdot f(l_d | \mu_r, \Sigma_r) \cdot (c_r + \alpha_r), \quad (4.1)$$

where $c^k = \sum_{w'} c_{*,w'}^k$ and $\bar{\beta} = \sum_{w'} \beta_{kw}$.

In this proposal, we sample a topic from $\frac{c_{*,w}^k + \beta_{kw}}{c^k + \bar{\beta}}$ and a region from $f(l_d | \mu_r, \Sigma_r) \cdot (c_r + \alpha_r)$ independently. For simplicity, we use $p(k, r)$ to denote $p(z^{(m,n)} = k, r^{(m,n)} = r | \Phi^{-(m,n)})$ and $q_{indep}(k, r)$ to denote $q_{indep}(z^{(m,n)} = k, r^{(m,n)} = r)$. The acceptance probability of the samples drawn from the proposal in Metropolis-Hastings is then computed as:

$$\min(1, \frac{p(k_t, r_t) q_{indep}(k_{t-1}, r_{t-1})}{p(k_{t-1}, r_{t-1}) q_{indep}(k_t, r_t)}).$$

The term used to sample topics, i.e., $\frac{c_{*,w}^k + \beta_{kw}}{c^k + \bar{\beta}}$, ensures similar word distribution of the topics as does the true posterior. Moreover, the term used to sample regions, i.e., $f(l_d | \mu_r, \Sigma_r) \cdot (c_r + \alpha_r)$, preserves the spatial proximity and the region popularity in the true posterior. The independent-proposal only needs to access the word-topic matrix and the region parameters in which the region parameters are often small ($O(R)$), and thus the communication cost is mainly from the word-topic matrix.

Joint-proposal. We propose the joint-proposal to capture the correlation between topic and region. It is defined as:

$$q_{joint}(z^{(m,n)} = k, r^{(m,n)} = r) \propto (m_{kr} + \gamma_{kr}), \quad (4.2)$$

with acceptance probability of the samples:

$$\min(1, \frac{p(k_t, r_t) q_{joint}(k_{t-1}, r_{t-1})}{p(k_{t-1}, r_{t-1}) q_{joint}(k_t, r_t)}).$$

The joint-proposal only accesses to the topic-region matrix and generates samples based on the correlation of topic and region without considering the current word token and document. It is used to make sure the learned topics and regions are correlated while the independent-proposal freely explores the solution space of the topic model and the Gaussian mixture model.

Note that we do not exclude the topic and region assignment of the current word token in the counts in the independent proposal, because the topic and region assignment to

a single word does not make much difference to the simulation using the proposals, but this treatment enables us to reduce the training complexity using Alias Table [LARS14], which will be presented in Section 4.2.4.

Compound proposal. Observation 3 indicates that we need to incorporate the two proposal distributions in order to preserve all the statistics in Equation (2.4). Hence, we combine the two proposals to a compound proposal by the cyclic method [Tie94], i.e., the two proposals are used alternatively to sample the topic-region pair for each word token. More specifically, after we sample the topic and region for a word token with the independent-proposal, we switch to use the joint-proposal to process the next word token. The cyclic method is proven to be able to converge and is equivalent to a probability distribution with density that is equal to the multiplication of the densities of two proposals [Tie94]. In our case, the multiplication of the two proposals results in a proposal distribution that is in the same form of Equation (2.4).

The compound proposal is able to support independent access to the word-topic and topic-region matrices for training geographical topic models. Because word token is processed by only one proposal each time, sampling the topic and region for a word token only needs to access either the word-topic matrix or the topic-region matrix.

4.2.2.2 Model Parallelism

Based on the proposed training algorithm, we design a model parallelism that is able to increase the parallelism and reduce the memory requirement of worker machines.

The idea of improving the parallelism while reducing the memory requirement of worker machines is to partition the two matrices into slices, and each time a worker machine only fetches and stores one slice for training. At the sample time the worker machine can pre-fetch the next slice to parallel the computation and communication. Figure 4.2 illustrates the proposed model parallelism.

Before training, we allocate the two proposals alternatively to the word tokens. We partition the word-topic matrix by words into word slices (blue rectangle in Figure 4.2) because the words are observable in the text and we know which words correspond to the word slices such that we can only sample the topics and regions for the word tokens in the word slice. Similarly, we partition the topic-region matrix by regions into region slices (in red rectangle in Figure 4.2). To know the region of a word token, we propose a two-step sampling scheme for the joint-proposal in Section 4.2.3.2. Once the regions are known, we sample the topics for the word tokens that are assigned to the region in the region slice (e.g., r_1 and r_2 in the joint-proposal). In each training iteration, a worker fetches the parameters slice by slice from the parameter servers and samples the topics and regions for word tokens corresponding to the word/region slice. After using a word/region slice, the updates are pushed to the parameter servers and the local copy of the slice can be removed from memory.

Note that we only focus on the model parallelism for the word-topic matrix and the topic-region matrix since other parameters, i.e., Gaussian parameters μ_r, Σ_r , region counts c_r and topic counts c^k , are of small size and can easily fit into memory.

Figure 4.2: An example of model parallelism.

4.2.3 Parameter Localization

To reduce the communication cost, we propose two ideas for parameter localization. First, we propose a data parallelism that use spatial partitioning techniques to ensure documents in each worker is spatially close (Section 4.2.3.1), and thus the data of a worker is the data parallelism ensures that the data in each worker is spatially close and only related to several regions (but not all the regions). Then, we propose a two-step sampling scheme in Section 4.2.3.2 that is able to achieve on-demand accessing such that each worker only needs to access part of the region-topic matrix, i.e., matrix slices for regions that are related to the worker, thus reducing communication cost. Based on the first idea, the second idea is to design an approach such that the region-topic matrix slices are stored in the shared memory of related workers, and we present the idea in Section 4.2.3.3.

4.2.3.1 Data Parallelism

In all the previous work on distributed training of topic models, documents are partitioned randomly and evenly across different machines so that all the workers have similar workload. We can follow the same strategy. However, if the data in each worker machine are

spatially close, it may relate to only part of the latent regions and thus provides flexibility to reduce network communication cost. Consider both spatial proximity and workload balance, we propose to use spatial partitioning to distribute the data to worker machines. Several spatial partitioning techniques [AMH⁺15, XLY⁺16, YZG15, EM15, YWS16] can be used, e.g., k-d tree, R-tree, etc. Without loss of generality, we use k-d tree to partition the data because it is easy to implement and extend. We do not aim to compare different spatial partitioning techniques, and leave such comparison as future work.

K-d tree recursively partitions the space into two cells with equal number of documents from one of the two dimensions. We use k-d tree to partition the data to M partitions and dispatch each data partition to one machine. When a data partition in a machine cannot be fit in memory, we further partition it using k-d tree to get B data blocks, such that each of them can be fit into memory.

In each data block, we store the word token and corresponding topic and region assignments as tuples $\langle w_{mn}, z^{(m,n)}, r^{(m,n)} \rangle$, because they always co-occur during the training process. When the worker processes a word/region slice, the tuples correspond to the slice are processed. After all word tokens are processed, the tuples are written to disk and the next data block is loaded for training.

4.2.3.2 On-demand Fetching of Topic-Region Matrix

The model parallelism of the topic-region matrix requires to know the region assignment of each word token as illustrated in Figure 4.2. Once we know the regions, we can fetch the region slices that are related. Furthermore, if the data in a worker machine is spatially close, only a small subset of related region slices may be required for the worker machine. In other words, the worker machines will not need to access the whole topic-region matrix. Thus, the on-demand fetching not only supports the model parallelism but also reduces the communication cost. However, the proposed joint-proposal only relates to the topic-region matrix, which makes it difficult to sample a region without fetching the whole matrix from the parameter servers.

We proceed to propose a two-step sampling scheme that is able to achieve on-demand accessing to the topic-region matrix while preserving the correctness of sampling for learning models. The idea is to extend the joint-proposal by adding the Gaussian mixture term in Equation (2.4):

$$q_{joint}(z^{(m,n)} = k, r^{(m,n)} = r) \propto (m_{kr} + \gamma_{kr}) \cdot f(l_d | \mu_r, \Sigma_r) \cdot (c_r + \alpha_r), \quad (4.3)$$

in which the Gaussian density $f(l_d | \mu_r, \Sigma_r)$ captures the spatial proximity and $(c_r^{-d} + \alpha_r)$ represents the popularity of the region. With this extension, the joint-proposal can be viewed as a mixture model, i.e., the probability of sampling a topic k from it is

$$p(z^{(m,n)} = k) = \sum_r p(z^{(m,n)} = k, r^{(m,n)} = r) p(r^{(m,n)} = r),$$

where $p(r^{(m,n)}) \propto f(g_d|\mu_r, \Sigma_r) \cdot (c_r + \alpha_r)$.

Our two-step sampling scheme follows the sampling routine of a mixture model. More specifically, in the first step, we sample a region for a word token based on $f(l_d|\mu_r, \Sigma_r) \cdot (c_r + \alpha_r)$; In the second step, we sample a topic k according to the probability of observing region r together with topic k , which is proportional to $m_{kr} + \gamma_{kr}$. Because the Gaussian distribution has exponential decrement of density w.r.t. the distance, the regions that are far from a document are less likely to be sampled when sampling for word tokens of the document. We then only need to access the topic distribution of the sampled region from the topic-region matrix.

Note that extending the joint-proposal with the Gaussian mixture cause no extra or missing statistics in the posterior in Equation (2.4) and thus will not affect the mixing rate of the Metropolis-Hastings algorithm.

Algorithm 4 shows the sampling process with the proposed two-step sampling scheme in a worker machine. In Algorithm 4, we first assign one of the two proposals (independent or joint proposal) to each word token alternatively (Line 4). Then for each word token, a sequence of s (burn-in steps in Metropolis-Hastings sampling) samples of regions r is drawn from the posterior of the Gaussian mixture, i.e., $f(l_d|\mu_r, \Sigma_r) \cdot (c_r + \alpha_r)$ (Line 5). We then pull the word-topic slices from parameter servers to sample regions and topics with the independent proposal (Lines 8-10) and pull the related topic-region slices from parameter servers to sample with the joint proposal (Lines 11-13). Finally, the region assignment of the document is randomly select from one of its word's region assignment (Lines 14-16).

4.2.3.3 Parameter Allocation to Related Worker

The parameter servers store the global parameters in the shared memory, which is distributed over all the machines. Based on the data parallelism and the two-step sampling scheme, we can find the regions that a worker is related to, and the corresponding region-topic matrix slices. We propose the second idea to reduce communication cost, which stores region-topic matrix slices in the shared memory of the related worker, and then accessing these slides from the parameter servers only needs access local machine. We achieve this by initializing the Gaussian mixture parameters of the regions with the data in the same machine. For example in Figure 4.2, we initialize r_1 and r_2 with the documents in worker 1 such that r_1 and r_2 (depicted as red ovals) are spatially close to the documents in worker 1. We store r_1 and r_2 in the shared memory of worker 1, and thus worker 1 only needs to read from its memory to access the slices of global parameters for regions r_1 and r_2 .

Algorithm 4: Sampling process in a worker

```

begin
1   for  $i = 0$  to  $maxIter$  do
2       foreach data block  $D_b$  in the worker do
3           load  $z^{(m,n)}$  and  $r_m$  from disk ;
4            $q_{mn} \leftarrow$  Equation (4.1) or Equation (4.3) ;
5            $R_{mn} \leftarrow$  Sample  $s$  regions for each word token  $w_{mn}$  ;
6            $R_{demand} \leftarrow$  unique regions from  $R_{mn}$  ;
7           while Not all the tokens are sampled do
8               fetch a word-topic slice with words  $V_i$  ;
9               foreach  $m, n$  where  $w_{mn} \in V_i$  and  $q_{mn}$  is Equation (4.1) do
10                   $z^{(m,n)}, r^{(m,n)} \leftarrow$  Metropolis-Hastings( $p, q_{mn}, R_{mn}, s$ ) ;
11               fetch a topic-region slice overlaps  $R_{demand}$  ;
12               foreach  $m, n$  where  $r_{mn} \in R_{demand}$  and  $q_{mn}$  is Equation (4.2) do
13                   $z^{(m,n)}, r^{(m,n)} \leftarrow$  Metropolis-Hastings( $p, q_{mn}, R_{mn}, s$ ) ;
14               foreach  $m$  in  $|D|$  do
15                   $n' \leftarrow \text{random}(1, |\mathbf{w}_m|)$  ;
16                   $r_m \leftarrow r^{(m,n')}$  ;
17               store  $z^{(m,n)}$  and  $r_m$  to disk ;
18           if Converge then
19               break ;
    
```

4.2.4 Reducing Training Complexity

Apart from the three design objectives, the proposed training algorithm and the two-step sampling scheme reduce the training complexity from $O(KR)$ to $O(K + R)$ because sampling from either proposal takes $O(K + R)$. To further improve the training efficiency, we propose an approach to reduce the amortized complexity of sampling a topic for a word token from $O(K)$ to $O(1)$ and sampling a region for a word token from $O(R)$ to $O(1)$ based on Alias Table [LARS14].

Alias Table [LARS14] has been proposed to reduce the sampling complexity of a discrete distribution of K components to $O(1)$ instead of $O(K)$. Figure 4.3 illustrates how to transform a discrete distribution to an Alias Table. The idea is to borrow probability mass from components with high probability ($> 1/K$) for those with low probability ($< 1/K$) to transform the non-uniform distribution to uniform ones. The transformation runs at most K rounds. At each round, a low component ($< 1/K$) borrows the mass from one high component ($> 1/K$) to reach $1/K$ probability. For example in Figure 4.3, the number stands for the steps of the transformation. In the first step, the first

component borrows 0.15 from the second component to reach $1/K = 0.2$, and the second component becomes 0.1. In the second step, the second component borrows 0.1 from third component. The transformation ends with a uniform distribution in which each bin (component) consists of at most two components (in different colors) from the original distribution [MTW⁺04], and the probability mass of one of the two components is stored. To sample from the Alias Table, one bin in the Alias Table is randomly selected in $O(1)$ complexity, and then one of the two components in the same bin is sampled according to the probability mass.

Figure 4.3: An example of transforming a distribution to an alias table

While sampling from the Alias Table only takes $O(1)$ time, computing the probability and constructing the Alias Table costs $O(K)$. If the Alias Table can be reused for drawing K samples, then the construction complexity of Alias Table is amortized to $O(1)$.

Move back to our two proposals. Considering the sparsity of the matrices, the independent-proposal has a sampling complexity of $O(K_w + R)$ to sample the topic and region independently, where K_w is the number of topics assigned to word w in the word-topic matrix. The joint-proposal takes $O(|\psi|_{kr})$ to sample from the topic-region matrix with $|\psi|_{kr}$ non-zero entries. We build the Alias Table for $c_{*,w}^k$ and m_{kr} , and use the tables to draw K_w and $|\psi|_{kr}$ samples, respectively, before updating them. Thus, the complexity of sampling from the two proposals reduces to $O(R)$ and $O(1)$, respectively. Moreover, if the prior β_{kw} and γ_{kr} are set to non-uniform distributions, we can also use Alias Table to reduce the sampling complexity to $O(1)$.

Unfortunately, Alias Table cannot be applied to reduce the complexity of sampling regions because the Gaussian density, e.g., $f(l_d|\mu_r, \Sigma_r)$ is continuous and needs to be computed for each document. Our idea is to discretize the space into cells, and use cells to represent the documents within their boundaries. This is equivalent to convert the Gaussian distribution to a discrete distribution in which each cell is one component.

Spatial partitioning techniques (e.g., k-d tree) can be applied to discretize the space. Once the space is discretized, we build an alias table for each cell instead of each document and reuse the alias table to sample latent regions. The two proposals are then modified as:

$$q_{indep}(z^{(m,n)} = k, r^{(m,n)} = r) \propto \frac{c_{*,w}^k + \beta_w}{c^k + \bar{\beta}} \cdot f(g_d | \mu_r, \Sigma_r) \cdot (c_r + \alpha_r), \quad (4.4)$$

$$q_{joint}(z^{(m,n)} = k, r^{(m,n)} = r) \propto (m_{kr} + \gamma_{kr}) \cdot f(g_d | \mu_r, \Sigma_r) \cdot (c_r + \alpha_r), \quad (4.5)$$

where g_d is the center of the cell which the document d belongs to. When the number of documents in each cell is roughly R , the amortized complexity of sampling regions reduces to $O(1)$. Hence, the sampling complexity of either proposal can reduce to $O(1)$. Because the cell representation of documents is an approximation to the documents' locations, we show that the approximation is bounded in terms of KL-divergence.

Theorem 4: Let $p_l(r) = f(l | \mu_r, \Sigma_r) \cdot (c_r + \alpha)$ be the distribution of regions given a location l and c_g be length of the diagonal of the cell g . Given a document d with location l_d in a cell g with center l_g , the KL-divergence $KL(p_{l_d} || p_{l_g}) \leq \frac{1}{4} c_g^2 \max_r \text{tr}(\Sigma_r^{-1})$, where $\text{tr}(\cdot)$ is the trace of a matrix. $\square$

Proof sketch: For simplicity, we abuse the notations g and d to be the location of the cell and document in the cell, respectively. Let $f_r(\cdot)$ be the Gaussian density of region r and c_r' be $c_r + \alpha$. Then, we have the following KL divergence between the two distributions:

$$\begin{aligned} KL(p_d || p_g) &= \sum_r \frac{f_r(d) c_r'}{Z_d} (\log \frac{f_r(d)}{f_r(g)} + \log \frac{Z_g}{Z_d}) \\ &\leq \max_r \log \frac{f_r(d)}{f_r(g)} + \log \frac{\sum_{r'} \frac{f_{r'}(g)}{f_{r'}(d)} f_{r'}(d) n_{r'}}{\sum_{r'} f_{r'}(d) n_{r'}} \\ &\leq \max_r \log \frac{f_r(d)}{f_r(g)} - \min_r \log \frac{f_r(d)}{f_r(g)} \end{aligned} \quad (4.6)$$

where Z_d and Z_g are normalization terms of p_d and p_g , respectively. When $|R| \leq 1$, the KL divergence is 0. We now consider the case of $|R| > 1$.

Next, we infer the bound of $\log \frac{f_r(d)}{f_r(g)}$. Let $\hat{\mathbf{x}}_{dr} = \Sigma_r^{-\frac{1}{2}}(\mathbf{l}_d - \mu_r)$, $\hat{\mathbf{x}}_{gr} = \Sigma_r^{-\frac{1}{2}}(\mathbf{l}_g - \mu_r)$, the Euclidean distance between the location of document/cell and the region, multiplies the inverse of standard deviation of the region. Then,

$$2 \log \frac{f_r(d)}{f_r(g)} = \hat{\mathbf{x}}_{gr}^T \hat{\mathbf{x}}_{gr} - \hat{\mathbf{x}}_{dr}^T \hat{\mathbf{x}}_{dr}. \quad (4.7)$$

When $\hat{\mathbf{x}}_{gr}^T \hat{\mathbf{x}}_{gr} \geq \hat{\mathbf{x}}_{dr}^T \hat{\mathbf{x}}_{dr}$, if we denote $\Delta_r = \hat{\mathbf{x}}_{gr} - \hat{\mathbf{x}}_{dr}$ we have:

$$0 \leq \log \frac{f_r(d)}{f_r(g)} \leq \frac{1}{2} \Delta_r^T \Delta_r,$$

When $\hat{\mathbf{x}}_{gr}^T \hat{\mathbf{x}}_{gr} < \hat{\mathbf{x}}_{dr}^T \hat{\mathbf{x}}_{dr}$, we have

$$-\frac{1}{2} \Delta_r^T \Delta_r \leq \log \frac{f_r(d)}{f_r(g)} < 0.$$

Combine the two situations, we have:

$$-\frac{1}{2} \Delta_r^T \Delta_r \leq \log \frac{f_r(d)}{f_r(g)} \leq \frac{1}{2} \Delta_r^T \Delta_r. \quad (4.8)$$

Substitute Equation 4.8 to Equation 4.6, we have:

$$KL(d||g) \leq \max_r \Delta_r^T \Delta_r \quad (4.9)$$

Let $\mathbf{l}_g - \mathbf{l}_d = [a, b]^T$, and the variance matrix:

$$\Sigma_r = \begin{bmatrix} \delta_x^2 & \rho \delta_x \delta_y \\ \rho \delta_x \delta_y & \delta_y^2 \end{bmatrix}$$

Then, we have:

$$\begin{aligned} \Delta_r^T \Delta_r &= \frac{1}{1 - \rho^2} \left(\frac{a^2}{\delta_x^2} + \frac{b^2}{\delta_y^2} - \frac{2ab\rho}{\delta_x \delta_y} \right) \leq \frac{1}{1 - \rho^2} \left(\frac{a}{\delta_x} + \frac{b}{\delta_y} \right)^2 \\ &\leq \frac{1}{1 - \rho^2} \left(\frac{1}{\delta_x^2} + \frac{1}{\delta_y^2} \right) (a^2 + b^2) = \text{tr}(\Sigma_r^{-1}) (a^2 + b^2). \end{aligned} \quad (4.10)$$

When a and b reach their largest value, i.e., when the doc is at the corner of the cell, the KL-divergence reaches highest value. Denote c_g as the length of the diagonal of the cell. Then, $KL(d||g) \leq \frac{1}{4} c_g^2 \max_r \text{tr}(\Sigma_r^{-1})$. Note that Σ_r is the covariance matrix learnt from the data, which is not a function of the cell. Therefore, in the following method to discretize the space, we only concern about c_g^2 . The error bound of sampling in discretized space indicates that the smaller the area of the cells, the lower bound we can obtain.

4.3 Implementation

Parameter server. Our method can be built based on most of the existing parameter servers. We implement the proposed solution on top of an open-sourced¹ stale-synchronized parameter server called Bösen [HCC⁺13].

I/O and network cost. Because we sample a data block once a time and fetch parameter slices from parameter servers during sampling, apart from the computation cost we

¹<https://github.com/sailing-pmls/jbosen>

need to consider extra I/O and network communication cost. Benefiting from our proposed training algorithm and model parallelism, we are able to parallelize sampling the current data block with the current parameter slices, and the I/O of next data block and the communication cost of the next slice. The idea is to prefetch the next data block into memory and prefetch the next parameter slices while sampling the current data block, thus reducing the I/O cost and communication cost. The idea is illustrated in Figure 4.4. We maintain a sampling thread, an I/O thread and a model thread. The sampling thread gets data blocks from the I/O thread and parameter slices from the model thread. The I/O thread keeps reading data blocks and feeding them to the sampling thread. The model thread retrieves parameters from network. To overlap the I/O operation cost, when the sampling thread is working on a data block (e.g., block 1 in Figure 4.4), the I/O thread reads the next data blocks (e.g., block 2) into memory. To overlap the network communication cost, a region slice is pre-fetched from network while the sampling thread is working on a word slice using the independent proposal (e.g., q_{indep}).

Figure 4.4: Pipeline to hide I/O and network time cost.

4.4 Experiments

4.4.1 Experimental Setup

Datasets. We conduct experiments on two datasets, namely *Twitter* and *Web*. The Twitter dataset is collected from New York city users during 2015, which contains more than 136 million geo-tagged documents. To evaluate the proposed solution on a large dataset with long text, We collect the Web dataset from the Common Crawl data², which contains 22 millions web pages. Because the web pages do not have geo-tags, we

²<https://aws.amazon.com/public-datasets/common-crawl/>

randomly assign the web pages to a point-of-interest dataset SimpleGeo’s Places³ which contains 12 millions unique locations in United States. The statistics of the two datasets are reported in Table 4.2.

Table 4.2: Statistics of the datasets

	Twitter	Web
#docs	136.69 M	22.04 M
#tokens	755.80 M	1.30 B
vocabulary size	316,890	162,694
latitude range	[30.85, 40.92]	[18.32, 71.30]
longitude range	[-74.25, -73.70]	[-168.02, -64.96]

Distributed training solutions. we compare the proposed solution (PGeoTopic) with two baseline solutions as well as four variants of PGeoTopic:

- **Baseline:** The straightforward solution in Section 4.2.1.
- **Baseline+M:** We apply the model parallelism of LightLDA [YGH⁺15, LARS14] on the word-topic matrix to extend **Baseline**. This solution still has to retrieve the whole region-topic matrix at the beginning of each iteration.
- **PGeoTopic-P:** PGeoTopic without overlapping I/O, computation and network communication (Section 4.3).
- **PGeoTopic-O:** PGeoTopic without the on-demand parameter fetching of method proposed in Section 4.2.3.2.
- **PGeoTopic-L:** PGeoTopic without allocating the parameters to related workers as does Section 4.2.3.3.
- **PGeoTopic-S:** PGeoTopic that distributes data to worker machines randomly, but not based on spatial partitioning (Section 4.2.3.1).

We set the both numbers of topics and regions at 10,000 by default. The hyper parameters of all methods are set at the same values, i.e., $\alpha = 0.1$, $\beta = 10^{-3}/|V|$, and $\gamma = 10^{-3}/|K|$. We also place a normal-Wishart prior to the Gaussian distribution of each region for all methods, to avoid zero denominator when updating the region parameters μ_r, Σ_r .

Experimental environment. We setup the distributed training environment on an Amazon EC2 cluster. The cluster contains 20 machines (API Name “r4.2xlarge”), each of which has 27 compute units, 61 GB RAM and 1 Gbps network. The default number of machines use for training is 20 if not specified.

³<https://freegisdata.rtwilson.com/>

Objectives. (1) We evaluate the scalability of PGeoTopic with model size, number of machines and data size (Section 4.4.2); (2) We evaluate the time used for computation versus network communication to show the usefulness of overlapping the computation and network communication (Section 4.4.3); (3) We evaluate the memory cost of worker machines and the network communication cost (Section 4.4.4 and Section 4.4.5).; (4) We evaluate the effectiveness and efficiency of the algorithm for reducing training complexity in Section 4.4.6; (5) We conduct a case study of the learnt regions and topics in Section 4.4.7.

4.4.2 Scalability

We first show that PGeoTopic is able to handle a big model with 100,000 topics and 100,000 regions. We run PGeoTopic on 20 machines on the Twitter dataset and plot the likelihood curve over time in Figure 4.5. Figure 4.5 shows that the model converges in 20 hours, which is a large improvement over the baseline methods which cannot finish the training in a week.

Figure 4.5: Likelihood v.s. training time on 20 machines.

We next evaluate the scalability of PGeoTopic with model size, data size and number of machines.

Scalability with model size. We run the two baseline methods and PGeoTopic on 20 machines. To evaluate the scalability with number of topics, we fix the number of regions at 10,000 and vary the number of topics from $\{1000, 10,000, 100,000\}$. To evaluate the scalability with number of regions, we fix the number of topics at 10,000 and vary the number of regions from $\{1000, 10,000, 100,000\}$. The results of average throughput

Figure 4.6: Varying number of topics

Figure 4.7: Varying number of regions

(which is measured by the number of tokens sampled per second) for each iteration are reported in Figure 4.6 and Figure 4.7.

We observe that **PGeoTopic** is several orders of magnitude faster than the two baseline methods. The throughput of the two baseline methods degrades fast as the number of topics/regions increases. The two baseline solutions cannot finish after running for a week on large model, e.g., number of topics is 100,000, and we do not report their results when the number of topics or regions is 100,000. The baseline solutions are slow because the complexity of them is linear to the size of the model. **Baseline** scales worst because it needs to request the whole word-topic matrix and topic-region matrix before training, increasing the time for network communication.

Note that, the baselines have throughput around 1 token per second on the Twitter dataset and 0.1 token per second on the Web dataset, which is impractical to run for the remaining experiments. Therefore, we set both numbers of topics and regions for the baselines to 100 by default in the following experiments. Note that showing the baseline in the rest of the experiment is just to study the scalability of baselines, but not to compare with PGeoTopic—baseline methods train models with much fewer parameters, which is much less expensive.

PGeoTopic is less sensitive to the number of topics/regions compared to the two baselines. We observe that when the number of topics/regions increases, the throughput of PGeoTopic slightly degrades. This is because when the model is large, it becomes more difficult to overlap the network communication cost with the computation for training models.

Scalability with number of machines. To study the scalability of PGeoTopic with the number of machines, We run PGeoTopic, Baseline and Baseline+M on 12, 16 and 20 machines, respectively, and report the average throughput in Figure 4.8(a) and Figure 4.8(b). Both the number of topics and the number of regions of PGeoTopic are set at 10,000. For the two baselines, we set the numbers of topics and regions at 100, which is a much less expensive to train.

Figure 4.8 shows that the proposed solution scales near linearly to the growth of number of machines on both datasets. Even though the two baseline methods learn less parameters, they have extremely low throughput, i.e., less than 200 tokens per second on both datasets. Our solution is efficient and scalable in learning geographical topic models, because the proposed training algorithm and the model parallelism parallelizes the computation and network communication, and reduces the network communication cost. Moreover, the algorithm of reducing the training complexity also helps improve the training efficiency.

Scalability with data size. We vary the size of the data by randomly sampling 10%, 50%, and 100% of the data to evaluate the scalability of PGeoTopic with different data sizes. Both the number of topics and the number of regions of PGeoTopic are set at 10,000. For the two baselines, we set both the number of topics and the number of regions at 100. The average throughput of each iteration is reported in Figure 4.9(a) and Figure 4.9(b). PGeoTopic has higher throughput on larger dataset, because when the dataset is large, the worker machines spend more time for computation on each model slice. When the dataset is small, the time spent on sampling computation may not be able to overlap the network communication. In contrast, the two baselines do not display the property.

Figure 4.8: Varying number of machines

Figure 4.9: Varying data size

4.4.3 Computation v.s. Network Communication

To evaluate the computation (sampling) versus network communication, we run PGeoTopic and PGeoTopic-P on the Twitter dataset, and report the breakdown of the average training time per iteration for computation and network communication in Figure 4.10. Both the number of topics and the number of regions are set at 10,000.

Figure 4.10 shows that PGeoTopic-P is 1–3 times more expensive than PGeoTopic for communication cost while they take similar time for computation. The reason is that PGeoTopic-P does not overlap the communication cost and computation. Our experimental results demonstrate the benefits of overlapping the network communication and computation.

Figure 4.10: Training time breakdown - average time cost of network communication v.s. computation per iteration.

4.4.4 Memory Requirement on Worker Machines

We next evaluate the memory requirement on the worker machines. We run PGeoTopic on 12, 16 and 20 machines, and report the average memory usage for each machine during each iteration in Table 4.3.

Table 4.3: Memory usage of PGeoTopic in gigabytes

# Machines	12	16	20
Worker	7.92	7.51	7.44
Shared Memory	24.86	18.92	15.24
Total	32.79	26.43	22.70

The row “Worker” reports the amount of local memory used for training the model while “Shared Memory” of each worker machine is employed for parameter servers. The shared memory usage drops when the number of machines increases because the memory usage for model parameters is amortized to more machines. Table 4.3 shows that the worker memory usage remains stable when the number of machines increases. This is because our solution PGeoTopic only needs to load one model slice into memory each time. The experimental result also indicates that PGeoTopic is applicable to machines with limited memory.

4.4.5 Network Communication Cost

In this experiment, we evaluate the usefulness of the parameter localization of PGeoTopic that is designed to reduce the communication cost. Specifically, we compare PGeoTopic

with three variants — (1) **PGeoTopic-O** that disables the on-demand parameter fetching (i.e., using all of the region slices in each iteration); (2) **PGeoTopic-L** that assigns the region slices to the machines randomly without allocating parameters to related workers; (3) **PGeoTopic-S** that randomly allocates the data to workers without spatial partitioning. We run all the methods with the default number of topics and regions on 20 machines, and report the network communication time and the average size of the topic-region parameters transmitted via network for each machine in each iteration in Figure 4.11 and Figure 4.12, respectively.

PGeoTopic reduces the network communication time by 58.7% compared to the best variant, i.e., **PGeoTopic-L**. Both **PGeoTopic-S** and **PGeoTopic-O** have high communication cost. **PGeoTopic-S** does not apply spatial partitioning, and thus the on-demand parameter fetching method does not work properly. **PGeoTopic-O** requires to fetch all parameters in each iteration and thus performs worst. The result demonstrates the usefulness of both spatial partitioning and on-demand fetching. **PGeoTopic-L** has smaller communication cost compared with the other two variants, benefitting from the on-demand parameter fetching method. However, it still performs worse than **PGeoTopic**. This is because the proposed distributed training solution for geographical topic models achieves parameter localization by allocating the parameters to related workers, thus reducing the network communication cost significantly.

Figure 4.11: Average network communication time (in sec.) per iteration.

Figure 4.12: Average size of the topic-region parameters (in GB) transmitted via network per iteration.

4.4.6 Sampling Efficiency and Effectiveness in Single Machine

We study the efficiency and effectiveness of our algorithm for reducing the complexity on a single machine. We compare our method with Gibbs sampling and a variant of our

method (PGeoTopic-D) without discretizing the space. The complexity of PGeoTopic-D is $O(WRI)$. We conduct the experiment on two smaller datasets, GeoText and Yelp, because Gibbs sampling takes long time to run on a large scale dataset. The GeoText data is published by researchers from Carnegie Mellon University [EOSX10], which contains 171,487 geo-tagged microblog messages (1,072,210 word tokens) posted mainly in United States. Yelp dataset contains 215,124 reviews (14,842,054 word tokens) for points of interest in Phoenix which is downloaded from Yelp Dataset Challenge⁴. On the GeoText dataset, we set the numbers of topics and regions at 10 and 100, respectively, and run all the methods for 100 iterations. On the Yelp dataset, the Gibbs sampling method requires more than two days to train a model with 10 topics and 60 regions for 100 iterations. For comparison purpose, we set a smaller number of regions (e.g., 60) on the yelp dataset and run Gibbs sampling for 100 iterations. In order to observe the convergence of PGeoLDA and PGeoLDA-D on the Yelp dataset, we run the two methods for 500 iterations. The log-likelihood growth along iterations and time of the three methods (e.g., PGeoTopic, PGeoTopic-D and Gibbs sampling) are reported in Figure 4.13 and Figure 4.14. PGeoTopic-D is a variant of our method without discretizing the space, which has a training complexity of $O(WRI)$.

Figure 4.13: Log-likelihood growth along iterations

From Figure 4.13, we observe that the proposed sampling method achieves similar log-likelihood and converge as fast as Gibbs sampling and PGeoTopic-D on both datasets. The slight log-likelihood lost is caused by the approximated representation of document locations by discrete cells. On the other hand, Figure 4.14 shows that our proposed sampling method is much faster than Gibbs sampling, which cannot finish the training

⁴https://www.yelp.com/dataset_challenge

Figure 4.14: Log-likelihood growth along time

 Figure 4.15: Training time of different number of topics ($R = 10$)

in two days on the Yelp dataset. PGeoTopic is at least 2x faster than PGeoTopic-D because we reuse the alias table for documents in a cell.

Apart from the training effectiveness, we also study the efficiency of the three methods on different number of topics and regions, and report the results in Figure 4.15 and Figure 4.16. In Figure 4.15, the number of topics K varies from 20 to 100. We observe that the training time of both PGeoTopic and PGeoTopic-D is constant w.r.t. the number of topics because both methods build alias tables for the topic vectors of words to achieve $O(1)$ amortized complexity. The Gibbs sampling algorithm takes more than 5,000 seconds and 60,000 seconds to train 20 topics on GeoText and Yelp, respectively, and the training time grows linearly w.r.t. the number of topics. In addition, PGeoTopic and PGeoTopic-

 Figure 4.16: Training time of different number of regions ($K = 10$)

D are much faster than Gibbs sampling when the number of regions varies as shown in Figure 4.16. PGeoTopic achieves constant training time w.r.t. both the number of topics and the number of regions, and thus scale well on large dataset.

4.4.7 Case Study

We explore some example latent regions and topics in this section. In particular, we select 100 most informative regions from the 10,000 learned regions ranked by the entropy w.r.t. the topic distribution of each region. Regions have high entropy only mention some particular topics rather than covering all of them. Due to the space limitation, we further manually select four interesting regions from the 100 informative regions, and plot them on the map as shown in Figure 4.17. The top two topics and their representative words (with high probability) for each region are listed in Table 4.4.

Observe from Figure 4.17(a) and Figure 4.17(b), the regions of university and beach have different topics. The university area cover topics such as class and campus life, while the topics in beach area focus on entertainment (e.g., wearing, eating, and drinking). On the other hand, Figure 4.17(c) and Figure 4.17(d) shows that even in the same airport, the model can discover different topical-regions (or functional regions) such as terminal gates and waiting lounges. As observed in Figure 4.17(c) and Figure 4.17(d), the two regions in airport have different topics. Topics in waiting lounges are mainly about things to do for time killing and flight delays, while in the terminal area, people often talk about arrival/departure and carrier information.

Figure 4.17: Four example regions learned on the Twitter dataset. The blue ovals are the contour lines of the Gaussian distributions with confidence level $p = 0.9$. The red pins are the centers of the regions.

4.5 Summary

In this chapter, we study the problem of distributed training of geographical topic models to increase parallelism, reduce the memory requirements and reduce the communication cost. We propose a novel training algorithm and model parallelism for geographical topic models that enable parallelism between computation and network communication while reducing the memory requirement of worker machines. We also propose to achieve parameter localization for reducing network communication with two ideas – on-demand parameter fetching and allocating parameters to related workers. In addition, to improve

Table 4.4: Topics for each region

Region	Topic ID	Representative words
University	4879	class, hit, sleep, caring, clothes, easier
	2987	campus, silent, weekend, pic, nights
Beach	612	beach, sweatshirt, remember, eat, hold
	2929	Drinking, drink, #Veterans, cheat, maybe
Airport - waiting lounges	958	urge, phone, @mangomeagan, Playing, fm
	7955	delayed, #Weather, fan, engaged, destined
Airport - terminal gate	9321	Uber, ending, #z100MendesMadness, ready, fam
	4428	airport, let, carrier, #cathedral, hashtag

the computation efficiency, we propose an algorithm to reduce the training complexity. Our experimental results demonstrate that the proposed distributed training solution is scalable w.r.t. number of machines, data size and model size, and outperforms baseline solution by several orders of magnitude.

Chapter 5

Annotating POIs with Geo-tagged Microblogs

Most of recent studies [YCM⁺13, YZS⁺15, WYC⁺15, YCZ⁺15] on mining topics of individuals from check-in data which often contains limited textual information. Moreover, most of the users have few check-ins. To provide a complementary check-in source to the traditional location-based services for mining topics of individuals, we first study the research problem of annotating POIs with geo-tagged microblogs. This work is different from the existing studies [WB11, HAG⁺12, DKP12, AHS13, LHFT14] on inferring the coordinates for documents in that we focus on geo-tagged microblogs and identify the related POIs of the microblogs. In addition, we consider that a geo-tagged microblog can be either POI-related or non-POI-related while the existing studies [LX11, SSSH13, BKB15] assume each document is POI-related and associate it to a POI. We propose a supervised Bayesian model that considers three factors: text content, coordinates and user behaviors under the supervision of POI-relatedness. Experiments show that the proposed method is effective in finding POI-related microblogs and associating them to related POIs.

This chapter is organized as follows. Section 5.1 defines the problem of annotating POIs with geo-tagged microblogs. Section 5.2 introduces two straightforward solutions for the problem and Section 5.3 presents the proposed supervised Bayesian model. Section 5.4 reports the experimental study and Section 5.5 summaries this chapter.

5.1 Problem Definition

Suppose we have a collection of historical records of microblogs¹ associated with their POIs, denoted by $D = \{d_1, d_2, \dots, d_{|D|}\}$. Each historical record d is represented as a quadruple $\langle u_d, \widetilde{\mathbf{cd}}_d, \mathbf{w}_d, l_d \rangle$, where u_d , $\mathbf{cd}_d$, $\mathbf{w}_d$ are the user, coordinates, and the set of

¹We use tweets and microblogs interchangeably in the remaining of this chapter.

words of tweet d , respectively, and l_d represents the POI where the tweet was posted or associated with. Attribute l_d is *NULL* if the tweet is not related to any POI. We also have a collection of POIs $L = l_1, l_2, \dots, l_{|L|}$ and each POI l is represented as a pair $\langle \widetilde{\mathbf{cd}}_l, \mathbf{t}_l \rangle$, where $\widetilde{\mathbf{cd}}_l$, $\mathbf{t}_l$ are the coordinates and the text context of the POI. The text context contains the name of the POI as well as the tips of the POI². Given a new tweet $\langle u_d, \widetilde{\mathbf{cd}}_d, \mathbf{w}_d \rangle$, the POI annotation problem is to *EITHER* return the top-1 POI that is relevant to the tweet if it is POI-related, *OR* return no POI if it is not-POI-related.

5.2 Basic Solutions

This is a new research problem and there is no existing solution. Since we have both POI related and unrelated tweets, directly annotating the top-ranked POI to each tweet only according to distance or text similarity performs bad in our experiments. Hence, we propose two basic solutions based on both spatial and textual information. The first one is a ranking based model which combines a spatial filter and a ranking component based on language models. The second one solves the problem in two steps: First, we classify the tweets into two classes, namely POI-related and non-POI-related; Second, a ranking model is used to map a POI-related tweet to a POI based on both the language model and spatial distance. The two models are named as **RANK** and **CLASS**, respectively.

RANK: Intuitively, if a tweet is posted at a POI, the coordinates of the tweet and the POI should be close to each other. Note that their coordinates are seldom the same because of the accuracy of GPS devices and the spatial region of a POI (which may not be a single coordinate point). The idea of **RANK** is to apply a spatial filter to restrict the search area for a tweet to be associated, and then to map the tweet to a POI in the restricted area. Suppose we restrict the search space within a small range of nearby POIs (e.g., 100 meters) centered at the tweet’s coordinate. The next problem is to find out whether any POI in the search space is semantically close to the tweet. More specifically, let $L_{d,m}$ be the set of POIs within m meters of tweet d , and $P(\mathbf{w}|l)$, $P(\mathbf{w}|d)$ be the language models of POI l and tweet d , respectively. The language model for a POI l is computed by counting the words in the text context of the POI (i.e., $\mathbf{t}_l$), and the words in the historical tweets posted in the POI i.e., $\cup\{\mathbf{w}_d | l_d = l\}$.

RANK associates tweet d to a POI $l \in L_{d,m}$ that achieves the highest likelihood of the tweet as follows:

$$\arg \max_{l \in L_{d,m}} \prod_{w \in \mathbf{w}_d} P(w|l).$$

Note that, this ranking mechanism will associate a geo-tagged tweet to a POI even though it is non-POI-related. To solve this problem, we add a dummy POI which stands

²The name and tips are collected from Foursquare in our work.

for “Non-POI”, and build a location independent language model for it using all non-POI-related tweets in the training set. We rank all the POIs together with the dummy POI and pick the top-1 result. If the dummy POI is the top-1 result, then the tweet is considered to be non-POI-related.

CLASS: This method first classifies the tweets into POI-related or non-POI-related, and then maps POI-related tweets to their corresponding POIs based on the combination of a distance model and a language model. To classify tweets into the two classes, we use the words in the training tweets as features to learn a linear Support Vector Machine (SVM).³

If a tweet is POI-related, the next problem is to map the tweet to its POI. We develop a ranking model which comprises a distance model and a language model to identify the POI for POI-related tweet d . The distance model is used to constrain the POI to be close to the tweet. To this end, we use a zero-mean normal distribution to model the probability of observing POI l given tweet d in spatial within an error tolerance σ^2 , i.e., $\text{dist}(d, l) \sim \mathcal{N}(0, \sigma^2)$. The variance σ^2 can be learnt on the history tweets using the maximum likelihood principle. The language model is built in the same way as we do for RANK. The top-1 ranked POI for tweet d is based on the overall ranking score computed by:

$$\text{Score}(d, l) \propto \exp\left\{-\frac{\text{dist}(d, l)^2}{2\sigma^2}\right\} \times \prod_{w \in \mathbf{w}_d} P(w|l).$$

Expanded CLASS: We now expand the CLASS model by incorporating region information in the classification. This model is named CLASS-R. Specifically, we first use k-means clustering to group the training tweets to R regions and assign the closest region to each tweet. Then, we compute the probability of posting POI related tweets $p(d \in D_+|r)$ in each region r , and the probability for a user u to posting tweets in each region $p(r|u)$, where D_+ is the set of POI-related tweets. We use the probability $p(d \in D_+|u) = \sum p(d \in D_+|r)p(r|d)$ as region features, and thus we have R region features for each tweet. For training tweets, we set $p(r|d) = 1$ if r is assigned to tweet d ; otherwise we set it at 0. For test tweets, we set $p(r|d) = p(r|u_d)$ for each region to smooth the region features using the user’s interests to regions.

For the ranker, we use two alternative ranking models. One of them is a learning-to-rank approach proposed by Lian et al. [LX11], and the other is a geographical topic model proposed by Yuan et al. [YCM⁺13]. In the learning-to-rank method, the following features are used: 1) number of check-ins at a POI; 2) check-in time of the POI; 3) the check-ins of a user at the POI; 4) the distance between the POI and the tweet; and 5) text similarity between the POI and the tweet. We name the learning-to-rank method as CLASS-LR and the geographical topic model as CLASS-W4, based on the classification results of CLASS. We also name the two ranking methods based on the results of CLASS-R as CLASS-R-LR and CLASS-R-W4, respectively.

³We have also tried other classifiers and SVM performed the best in our experiments.

5.3 Supervised Bayesian Model

We now discuss the motivations of building a supervised Bayesian model for the proposed problem and present the generative story of our model. Then, we present the method for estimating model parameters. Our model is further enhanced by incorporating external textual content of the locations, i.e., Foursquare tips.

5.3.1 Motivation & Novelty

Both baselines presented in Section 5.2 cannot capture the relationships among variables. Next, we illustrate two example relationships and more relationships are presented as Intuitions in Section 5.3.2. For instance, there should be relations between mapping a tweet to a POI and determining its POI-relatedness. If we know that a tweet was probably posted at a bakery, then by comparing the location and language models of the tweet and the bakery, it is easier to tell whether the tweet is POI-related. There also should be relations among user, region and POI-relatedness. Some regions (e.g., Times Square), may have more POI-related tweets than other regions. If a user often visits POIs at Times Square, the tweets posted by the user at Times Square are more likely to be POI-related.

To better capture the relations among variables (e.g., text, region, user, location, and POI-relatedness), we develop a novel supervised Bayesian model. Compared with the existing geographical topic models [HAG⁺12, KIH⁺13, HE13, YCM⁺13, ZCYZ15], the novelty of our model is two-fold. First, we add a *relatedness response* to each tweet in our supervised model to determine whether the tweet is POI-related by considering the current context. The context includes 1) the number of words that are not related to any POI; 2) the distance from the tweet to its most semantically related POI; and 3) the latent region in which the tweet was posted. By jointly modeling words and latent regions with the relatedness response, we are able to find better latent regions and language models for solving the relatedness problem. Second, we introduce a set of dummy POIs to model the location of tweets that are not related to any POI. The dummy POIs benefit the model in that we can model the two types of tweets in a consistent way because the non-POI-related tweets can now be assigned to a “dummy POI”. Moreover, each dummy POI has its own language model and coordinates like real POI, capturing semantic and geographical information. Such information makes it possible to analyze the language models and popularity (in regions) of dummy POIs. To the best of our knowledge, the existing *geographical topic models* are all unsupervised, and none of them exploits the two novel aspects.

Our model works as follows. When judging whether a tweet is POI-related, the model first estimates the most probable POI (including dummy POIs), and then computes the

relatedness response. If the tweet is POI-related, the candidate POIs are ranked based on their textual and spatial information, and the joint probability of the tweet being POI-related and posted at each POI.

Compared with supervised LDA [BM07], our model associates several features in different scopes to the response including proportion of POI-related words in the scope of word tokens, geographical distance and latent region in the scope of document. We design a feature vector that contains different scopes of features for each tweet and the feature vector affects the training of both textual and geographical aspects of the model. Supervised LDA uses average topic assignments of a document as features for regression and does not consider different types of features.

5.3.2 Generative Story for Tweets

The supervised Bayesian model for annotating POIs with tweets are built according to the following intuitions.

Intuition 1: A user u may post geo-tagged tweets in some preferred latent regions, e.g., shopping streets, sightseeing areas. Each region contains a set of nearby POIs. User interests in regions can be described by a categorical distribution $p(r|u)$.

Intuition 2: To visit a POI in a region, user would consider both its popularity and distance. For a region r , we use a popularity distribution $p(l|r)$ and a bivariate Gaussian distribution over coordinates $p(\tilde{\mathbf{cd}}_l|r)$ to depict the two factors, respectively.

Intuition 3: A tweet may or may not be related to a POI. For POI-related tweets, the location and text are determined by the POI where they were posted. For daily conversational tweets that are not related to any POI, we can imagine that their location and text are determined by some dummy POIs that do not exist in the real world. We introduce a set of dummy POIs L' to model such tweets.

Intuition 4: In a POI-related tweet, users tend to use words related to the POI other than general words. That is, if the tweet is POI-related, the user may select words from both the word distribution of the POI $p(w|\eta_l)$ and background word (e.g., I, like) distribution $p(w|\eta_0)$. If the tweet is non-POI-related, the user may select words from the word distribution of a dummy POI and the background word distribution.

Intuition 5: If a tweet is POI-related, it is likely that the tweet is close to the POI where it was posted, and the text of the tweet is related to its POI. Moreover, if a tweet is posted in a region that has many attractions, e.g., sight seeing spots, restaurants, etc., it would be more likely to be POI-related other than tweets posted in regions that have fewer attractions. We use $\bar{x}_d$ to describe the portion of words that are related to the tweet's POI in tweet d , and $\text{dist}(d, l_d)$ to denote the surface distance between tweet d and its POI l_d . For each region, we use $\bar{c}_r$, which is the average count of POI-related tweets in region r against non-POI-related tweets to denote the probability of posting

Figure 5.1: Supervised Bayesian Model for POI Annotation

POI-related tweets in that region. Let R be the total number of latent regions, we have $R + 2$ features to determine whether a tweet is POI-related or not.

With the above intuitions, we proceed to describe the generative process of the proposed **sBM** model. For convenience, we show the graphical representation of the model in Figure 5.1 and summarize the notations in Table 5.1.

To generate a geo-tagged tweet, a user first visits a region r according to a multinomial distribution over user's interests $Multi(\theta_u)$. Then, the user randomly chooses a POI (which can be a dummy POI) to visit according to 1) a multinomial distribution $Multi(\phi_r)$ over its popularity in region r ; and 2) a bi-variate Gaussian distribution $\mathcal{N}(\mu_r, \Sigma_r)$. When the user compose the tweet, she would choose words either from the language model η_l of the POI (including dummy POIs) or from a background language model η_0 . We introduce a latent variable x for each word token in a tweet to identify from which language model the word is selected from. More precisely, if $x = 1$, the user selects the word from the language model of the POI (including dummy POIs). If $x = 0$, the user selects the word from the background language model. The coordinates of the tweet are generated according to a Gaussian distribution $\mathcal{N}(\widetilde{\mathbf{cd}}_l, \sigma_l^2)$ with error σ_l^2 .

Table 5.1: Summary of notations

Notation	Description
D_u	the set of tweets posted by user u
θ_u	the region interests of user u
r_d	latent region of tweet d
ϕ_r	the POI popularity in region r
$\mathbf{cd}_l, \sigma_l^2$	the location and variance in coordinates system of POI l
μ_r, Σ_r	the mean and variance in coordinates system of region r
η_0	background word distribution
η_l	word distribution for POI l
$x_{(d,n)}$	the switch of selecting word (d, n) from either background or POI language models
$\mathbf{cd}_d$	the coordinates of tweet d
κ_d	distribution of switch for document d
ω	the weights for the regression of POI-relatedness
σ^2	the variance of POI-relatedness

Now we come to the supervised part of our model. The label y , indicating whether a tweet is POI-related, comes from a linear regression model. The covariates (features) of the regression model comprise the portion of POI-related words $\bar{x}_d$, the distance between the tweet and the POI $\text{dist}(d, l_d)$, and the popularity of real POIs in the region $\bar{c}_r$. These covariates comprise vector $\bar{z}_d$, which is a $R + 3$ dimensional vector that comprises a constant 1 and the $R+2$ features, i.e., $\bar{z}_d = \langle 1, \bar{x}_d, \text{dist}(d, l_d), \bar{c}_1, \dots, \bar{c}_R \rangle$. The regression coefficients on these covariates comprises vector ω , which is a $R + 3$ dimensional vector containing the weights to the $R+2$ features, i.e., $\omega_1, \dots, \omega_{R+2}$ and the bias ω_0 . We generate a label $y \in \mathbb{R}$ of POI-relatedness according to a normal distribution $\mathcal{N}(\omega^T \bar{z}_d, \sigma^2)$ with variance σ^2 . The generative process are summarized as follows.

- For each user u ,
 - Draw a region preference distribution $\theta_u \sim \text{Dir}(\alpha)$
- For each region r ,
 - Draw a POI distribution $\phi_r \sim \text{Dir}(\gamma)$
- For each POI l including dummy POI,
 - Draw a word distribution $\eta_l \sim \text{Dir}(\beta)$
- Draw a background word distribution $\eta_0 \sim \text{Dir}(\beta_0)$
- For each tweet d posted by user u ,
 - Draw a switch distribution $\kappa_d \sim \text{Beta}(\rho)$
 - Draw a region $r_d \sim \text{Multi}(\theta_u)$

- Draw a POI $l_d \sim \text{Multi}(\phi_r) \times \mathcal{N}(\widetilde{\mathbf{cd}}_l | \boldsymbol{\mu}_r, \boldsymbol{\Sigma}_r)$
- Draw coordinates $\widetilde{\mathbf{cd}}_d \sim \mathcal{N}(\widetilde{\mathbf{cd}}_{l_d}, \sigma_l^2)$
- For each word position n in d ,
 - * Draw a switch $x_{(d,n)} \sim \text{Binomial}(\kappa_d)$
 - * If $x = 1$, draw a word $w_{(d,n)} \sim \text{Multi}(\eta_{l_d})$
 - * If $x = 0$, draw a word $w_{(d,n)} \sim \text{Multi}(\eta_0)$.
- Draw $y \sim \mathcal{N}(\omega^T \bar{z}_d, \sigma^2)$

5.3.3 Parameter Learning

An open problem is how to determine the locations of dummy POIs in order to generate the coordinates of non-POI-related tweets from a Gaussian distribution ($\mathcal{N}(\widetilde{\mathbf{cd}}_{l_d}, \sigma_l^2)$). It is possible to learn the locations of dummy POIs together with the other parameters in the model. However, by considering that 1) it could be time-consuming when the number of dummy POIs becomes large and 2) the locations of dummy POIs do not affect the model much because they have no exact geographical meanings, we adopt an approximate strategy to compute the Gaussian distribution for dummy POIs. Suppose we need to generate a set of dummy POIs with size $|L'|$, our goal is to divide the non-POI-related tweets into $|L'|$ subsets, such that the tweets are equally distributed in these subsets, and the POIs in each subset are geographically close to each other. This is because we want to avoid the extreme case that a large number of tweets are assigned to very few dummy POIs. Specifically, we iteratively divide the coordinates space into four equal-sized cells and build a Quadtree [SW85]. In each iteration, we divide the cell with largest number of tweets in the Quadtree. The dividing process stops when we have more than $|L'|$ leaf cells each contains a reasonable number of tweets (e.g., at least 100 tweets). Then, we pick the top $|L'|$ leaf cells with largest number of tweets as dummy POIs. The coordinates of a dummy POI is computed by averaging the coordinates of its tweets. Finally, the tweets that are not located in any of the top $|L'|$ cells are assigned their closet dummy POIs. The error σ_l of each POI including dummy POIs is then computed by regression:

$$\sigma_l^2 = \frac{1}{|D_l| - 1} \sum_{d \in D_l} \|\widetilde{\mathbf{cd}}_d - \widetilde{\mathbf{cd}}_l\|^2, \quad (5.1)$$

where D_l is the set of tweets assigned to POI l .

With the known Gaussian distributions of POIs and dummy POIs, the learning problem becomes to compute the other parameters by maximizing the corpus level likelihood. The likelihood of generating the corpus D using our model with the set of parameters Φ is computed by:

$$\begin{aligned} P(D|\Phi) &= P(y_d|\bar{z}_d) \prod_d P(r_d|u) P(l_d|r_d) P(\widetilde{\mathbf{cd}}_l|r_d) P(\widetilde{\mathbf{cd}}_d|l_d) \\ &\times \prod_d \prod_n P(w_{d,n}|x_{d,n}, \eta_0, \eta_{l_d}) P(x_{d,n}|\kappa_d), \end{aligned} \quad (5.2)$$

$$P(w_{d,n}|x_{d,n}, \eta_0, \eta_d) = \begin{cases} P(w_{d,n}|\eta_0) & \text{if } x_{d,n} = 0 \\ P(w_{d,n}|\eta_d) & \text{if } x_{d,n} = 1. \end{cases} \quad (5.3)$$

Estimating the parameters by maximizing the likelihood is intractable. We therefore develop a two-step learning algorithm combining Gibbs sampling and Expectation-Maximization algorithm to estimate the parameters.

Expectation: We approximate the posterior distribution of latent variables x and r given other variables using collapsed Gibbs sampling. With each x and r known, the Gaussian parameters μ_r , Σ_r and the regression parameters ω and σ^2 are updated in the maximization step. The Gibbs samplers for latent variable x and r are derived as in Equation (5.4) and Equation (5.5) (Equations are listed on the next page).

$$\begin{aligned} P(x_{(d,n)} = 0 | X_{-(d,n)}, D) &\propto (c_{*,*,d}^{0,-(d,n)} + \alpha) \frac{c_{*,w_{d,n},*}^{0,-(d,n)} + \beta_{w_{d,n}}}{\sum_v c_{*,v,*}^{0,-(d,n)} + \beta_v} \times \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(y_d - \omega^T \bar{z}_d)^2}{2\sigma^2}\right\} \\ P(x_{(d,n)} = 1 | X_{-(d,n)}, D) &\propto (c_{*,*,d}^{1,-(d,n)} + \alpha) \frac{c_{l_d,w_{d,n},*}^{1,-(d,n)} + \beta_{w_{d,n}}}{\sum_v c_{l_d,v,*}^{1,-(d,n)} + \beta_v} \times \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(y_d - \omega^T \bar{z}_d)^2}{2\sigma^2}\right\} \end{aligned} \quad (5.4)$$

$$P(r_d = r | R_{-d}, D) \propto (c_{u_d,*}^{r,-d} + \alpha) \frac{c_{*,l_d}^{r,-d} + \gamma_{l_d}}{\sum_l c_{*,l}^{r,-d} + \gamma_l} \times |\Sigma_r|^{-\frac{1}{2}} \exp\{(\widetilde{\mathbf{cd}}_d - \mu_r)^T \Sigma_r^{-1} (\widetilde{\mathbf{cd}}_d - \mu_r)\} \quad (5.5)$$

$$\begin{aligned} P(l_d = l | u_d, \widetilde{\mathbf{cd}}_d, \mathbf{w}_d) &\propto P(l, \widetilde{\mathbf{cd}}_d, \mathbf{w}_d | u_d) \\ &= P(\widetilde{\mathbf{cd}}_d | l) \sum_r \phi_{r,l} \theta_{u_d,r} P(\widetilde{\mathbf{cd}}_l | r) \prod_n \kappa_d \eta_{0,w_{d,n}} + (1 - \kappa_d) \eta_{l,w_{d,n}}. \end{aligned} \quad (5.6)$$

The variable $c_{l,v}^{x,-(d,n)}$ in Equation (5.4) is the count of assigning x to the word v in the tweets posted in POI l , excluding the assignment for the n -th word in document d . We use $*$ to denote ANY. For example, $c_{*,v}^{x,-(d,n)}$ stands for the count of assigning x to the word v in the tweets posted in ANY POI, excluding the n -th word in document d . Similarly, the variable $c_{u,l}^{r,-d}$ in Equation (5.5) is the count of assigning r to the tweets that posted by user u at POI l , excluding the assignment in document d .

Maximization: We update the Gaussian parameters μ_r , Σ_r and the regression parameters ω , σ^2 by maximizing the likelihood given the samples of x and r . Let D_r be the set of tweets that are assigned to region r , and $\widetilde{\mathbf{cd}}_{l_d}$ be the coordinates of the POI l_d

associated with tweet d . The update functions of the Gaussian parameters are computed as in Equation (5.7) and Equation (5.8).

$$\mu_r = \frac{1}{|D_r|} \sum_{d \in D_r} \widetilde{\mathbf{cd}}_{l_d} \quad (5.7)$$

$$\Sigma_r = \frac{1}{|D_r|} \sum_{d \in D_r} (\widetilde{\mathbf{cd}}_{l_d} - \mu_r)(\widetilde{\mathbf{cd}}_{l_d} - \mu_r)^T. \quad (5.8)$$

We update the regression parameters by computing $\bar{z}_d$ for each tweet d using the samples of POI-relatedness indicator x for each word, and region r . Specifically, we set $\bar{x}_d$ by computing the proportion of words that are POI-related. For the region features, since we know the region of a tweet is r_d by sampling, we set the value of the r_d -th region feature in $\bar{z}_d$ to $\bar{c}_{r_d}$, the proportion of POI-related tweets in region r_d , and set the values of other regions to 0. Let $\bar{Z}$ be the a $|D| \times (R + 3)$ matrix in which each row is $\bar{z}_d$ for a tweet d , the regression parameters are updated in Equation (5.9) and Equation (5.10).

$$\omega \leftarrow (\bar{Z}^T \bar{Z})^{-1} \bar{Z}^T y \quad (5.9)$$

$$\sigma^2 \leftarrow \frac{1}{|D|} (y^T y - y^T \bar{Z} (\bar{Z}^T \bar{Z})^{-1} \bar{Z}^T y) \quad (5.10)$$

The expectation step and maximization step are repeated until the likelihood converges. The latent variables can be efficiently sampled. Sampling x and r only needs to scan the tweet set D once and thus the complexity of sampling is $O(|W| + R|D|)$, where W is all word tokens in D and R is the number of regions. The update of parameters μ_r and σ_r has a summation on D and thus the complexity is $O(R|D|)$. For the update of regression parameters ω and σ^2 , the most time consumption part is the matrix multiplication of $\bar{Z}^T \bar{Z}$ and its inversion. The complexity of updating the regression parameters is $O(R^2|D| + R^3)$. Suppose we run I iterations for the estimation process, the overall complexity of the learning process is $O(I(|W| + R^2|D| + R^3))$.

5.3.4 Prediction for New Tweets

After learning the parameters, the POI of a tweet is predicted by the model in two steps. Given a new tweet $d = \langle u_d, \widetilde{\mathbf{cd}}_d, \mathbf{w}_d \rangle$, the POI-relatedness y_d is first predicted and then the POIs are ranked according to the joint probability $P(y_d = 1, l_d = l | u_d, \widetilde{\mathbf{cd}}_d, \mathbf{w}_d)$.

POI-Relatedness: In this step, we first compute the probability $P(l_d = l | u_d, \widetilde{\mathbf{cd}}_d, \mathbf{w}_d)$ as in Equation (5.6) to guess the most probable POI (including dummy POIs) for the tweet. The corresponding $y_d = \omega^T \bar{z}_d$ is then computed to determine whether the tweet is POI-related. If $y > 0$, we classify the tweet as POI-related.

In Equation (5.6), the estimation of variable κ_d is intractable. A collapsed Gibbs sampler similar to Equation (5.4) is used here without involving the POI-relatedness

label y . Specifically, we infer the following Gibbs sampler to sample the indicator x_n for each word $w_{d,n}$:

$$\begin{aligned} P(x_{(d,n)} = 0 | X_{-(d,n)}^{new}, \eta_0) &\propto (c_{*,*,d}^{0,-(d,n)} + \alpha) \times \eta_{0,w_{d,n}} \\ P(x_{(d,n)} = 1 | X_{-(d,n)}^{new}, \eta_l) &\propto (c_{*,*,d}^{1,-(d,n)} + \alpha) \times \eta_{l,w_{d,n}}, \end{aligned} \quad (5.11)$$

where $X_{-(d,n)}^{new}$ is the set of assignments of x for word tokens in the new document, and η are the language models learnt from the training process. Other variables are defined in the same way as in the training process. After sampling x for each word, we compute κ_d as $\frac{\sum_{n=0}^{N_d} I(x_n=1) + \rho}{N_d + 2\rho}$, where $I(\cdot)$ is an indicator function, N_d is the total number of words in tweet d , and ρ is used for smoothing.

POI Ranking: When a tweet is POI-related or $y_d > 0$, we compute the most probable real POI for the tweet by considering two factors. One is the posterior probability of the POI given in Equation (5.6), the other is the probability of observing the POI-relatedness label $y_d = 1$ given the POI l . In other words, we rank all the POIs in the descending order of the joint probability $P(y_d = 1, l_d = l | u_d, \widetilde{\mathbf{cd}}_d, \mathbf{w}_d) = P(y_d = 1 | \widetilde{\mathbf{z}}_{d,l}) P(l | u_d, \widetilde{\mathbf{cd}}_d, \mathbf{w}_d)$ and then return the top-1 ranked POI for the given tweet. The vector $\widetilde{\mathbf{z}}_{d,l}$ is the feature vector when we assign POI l to tweet d .

5.3.5 Incorporating Tips

Because a tweet has a limit of 140 characters, and if a POI has very few posted tweets, the language model built for the POI would be extremely sparse. With the sparse language model, it is difficult to correctly judge whether a tweet is related to the POIs. However, we can make use of the external context for POIs. Example external context could be Foursquare tips, Yelp reviews, etc. In this work, we focus on making use of Foursquare tips, but the model can accommodate other text resources.

Suppose we are given a set of tips D_t for all POIs. Each tip is represented as a pair of a POI l and a set of words $\mathbf{w}$, i.e., $\langle l, \mathbf{w} \rangle$, where l is the POI of the tip, and $\mathbf{w}$ is the words of the tip. By observing that the proportion of background words used in tips and tweets are different, we use different distributions of drawing a switch variable (κ) for the two sources. We build a generative model with switch variable κ for the tips, and then use the learnt language model for each POI and background language model as prior for the language models in the supervised Bayesian model. Specifically, for each word in a tip, we first draw a switch variable x , then we draw the word from the language model of the POI if $x = 1$, otherwise we draw the word from the background language model. We apply Gibbs sampling to learn the language models from tips. Suppose the count of assigning a word v to a POI l in tips is $c_{l,v}^{tips}$, we substitute the new prior $\beta'_{lv} = c_{l,v}^{tips} + \beta_v$ in Equation (5.4) to learn the model presented in Section 5.3.2.

5.4 Experiments

We show the effectiveness of the proposed supervised Bayesian model (**sBM**) by experimenting on two real world datasets. We first discuss the experimental setup, including the dataset preparation and performance measures. Then we set the parameters, e.g., the number of dummy POIs and the number of latent regions, by empirically studying on a validation dataset. Finally, we compare the performance of **sBM** with baseline methods, and conduct an empirical study on the regions learnt by the model.

5.4.1 Experimental Setup

Dataset: We collect English tweets for experiments using Twitter API⁴ in two cities: New York city (NYC) and Singapore (SG). Specifically, a random sample of geo-tagged tweets posted by 2,393 users in New York city were collected from September 2010 to January 2015. For Singapore, a random sample of geo-tagged tweets from 9,978 users were collected from March 2014 to August 2014. The POIs from both cities together with their tips were collected using Foursquare API. In the rest of this section, we use NYC and SG for the abbreviations of the two datasets for simplicity.

From each of the two cities, we randomly selected some tweets and asked three annotators to annotate whether a tweet is POI-related. The groundtruth label is based on majority voting from the three annotators. Another annotator is then engaged to associate the POI-related tweets to their nearby POIs. Finally, we obtained 4,827 and 5,827 geo-tagged tweets with groundtruth labels for model validation and evaluation, for the two cities, respectively.

In order to acquire a much larger dataset for training the models, we approximate the groundtruth for training using the following rule-based strategy. When a user checks in a POI in Foursquare, there is an option to share the check-in on Twitter as a check-in tweet. Similarly, Instagram also allows users to share their posts on Twitter. We consider as positive samples the Foursquare check-in tweets, and the tweets shared through Instagram and are associated with POIs. The remaining tweets that do not satisfy these conditions are considered as negative samples. This treatment results in more than 210 thousands training tweets for the NYC dataset, and 380 thousands of training tweets for the SG dataset. The detailed statistics of the two datasets are reported in Table 5.2.

Evaluation Measures: To evaluate the performance of our models, we adopt the measures that are similar to precision and recall. The difference is that we have two levels of predictions here, i.e., whether a tweet is POI-related, and whether a POI-related tweet is correctly associated with its groundtruth POI. Only when the groundtruth POI is correctly predicted for a POI-related tweet, the prediction is considered to be true positive.

⁴<https://dev.twitter.com/rest/public>

Table 5.2: Statistics of the two datasets

	New York city			Singapore		
# users	2,393			9,978		
# POIs	482,480			321,985		
# tweets (train validation test)	212,954	2,429	2,398	385,270	2,926	2,901
# POI-related tweets (train validation test)	43,010	474	559	29,290	221	289
# tips per POI	2.38			1.40		
Vocabulary size	332,369			111,415		

Let TP be the number of true positive, TP_R be the number of times we classify a tweet as POI-related, FP be the number of times we incorrectly associate any POI to a non-POI-related tweet, FN be the number of times we predict *NULL* for a POI-related tweet, and TN be the number of times we predict *NULL* for a non-POI-related tweet.

The precision and recall are computed as:

$$\text{Precision} = \frac{TP}{TP_R + FP}, \quad \text{Recall} = \frac{TP}{TP_R + FN}.$$

We can also evaluate a model for its ability of judging whether a tweet is POI-related or not, using similar measures. To distinguish from the above precision and recall, we call them *relatedness precision* and *relatedness recall* (or R-Precision and R-Recall for short). These two measures are computed as:

$$\text{R-Precision} = \frac{TP_R}{TP_R + FP}, \quad \text{R-Recall} = \frac{TP_R}{TP_R + FN}.$$

Since there is a trade-off between precision and recall, we also compute the F1 score for the two sets of measures, i.e.,

$$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

We use F1 and R-F1 to denote the F1 scores computed from the two sets of Precision and Recall, respectively.

Methods to Compare: We compare our model (sBM) with the two baseline methods (i.e., RANK and CLASS) discussed in Section 5.2. We also implemented the learning-to-rank model and the geographical topic model W4 for comparison. In total, 9 methods are evaluated in our experiments, summarized in Table 5.3.

The parameters of all methods are set at the best value in terms of F1 score on the validation set. For RANK, we tried the distance threshold from 50, 100, 200 and 500 meters, and then select 100 meters, which performed best in terms of F1 score on both

Table 5.3: Summary of the 9 methods evaluated in our experiments. The features used by these methods are listed (U – User preference, R – Latent regions)

Method	Description	Features	
		U	R
RANK	RANK model with distance threshold = 100 meters	-	-
CLASS	CLASS model	-	-
CLASS-R	CLASS model enhanced by regions	-	✓
CLASS-W4	CLASS model + geographical topic model W4	✓	✓
CLASS-R-W4	CLASS-R model + geographical topic model W4	✓	✓
CLASS-LR	CLASS model + learning-to-rank	✓	-
CLASS-R-LR	CLASS-R model + learning-to-rank	✓	✓
sBM-T	supervised Bayesian model without using tips	✓	✓
sBM	supervised Bayesian model	✓	✓

datasets. Smaller threshold (e.g., 50 meters) results in low recall because the related POI may be filtered out, while larger value (e.g., 200 meters) introduces more POIs to rank and thus may decrease the precision. For W4, the number of topics are set at 100 and the number of personalized regions is set at 2. The classifiers in both **CLASS** and **CLASS-R** are implemented using Liblinear [FCH⁺08]. The learning-to-rank model was implemented using SVM^{rank} [Joa06].

5.4.2 Parameter Selection

We have two parameters for tuning in the supervised Bayesian model. One is the number of dummy POIs, and the other is the number of latent regions. We first investigate the effect of dummy POIs by fixing the number of latent regions. Then we fix the number of dummy POIs and select the number of latent regions.

Setting the Number of Dummy POIs: We fix the number of regions to 50. Figure 5.2 plots the precision, recall and F1 by varying the number of dummy POIs from 10 to 10,000 on the validation sets of the two cities, NYC and SG. Note that, the x-axis is in log-scale. Observe from the figures, the varying of the number of dummy POIs does not affect the F1 score much. On the NYC dataset, the performance slightly increases as the number of dummy POIs increases. On the SG dataset, when the number of dummy POIs increases to 10,000, precision increases significantly, with degradation in recall values. The increase in the number of dummy POIs leads to decrease of the distance from an arbitrary tweet to a dummy POI. As the result, the probability for assigning a tweet to a dummy POI increases, thus leading to the increase in the probability of classifying a tweet as non-POI-related.

Considering the changes in precision, recall, and F1, we set the number of dummy POIs to 10,000 on both datasets.

Figure 5.2: Effects of the number of dummy POIs (fixing the number of latent regions at 50)

Setting the Number of Regions: With the number of dummy POIs fixed in both datasets, we investigate the effect of the number of latent regions. The precision, recall and F1 by setting different numbers of regions are reported in Figure 5.3 on both datasets.

On the SG dataset, the precision increases and reaches its best value when the number of regions increases to 30. The precision becomes stable for larger number of regions. On the other hand, the recall decreases slightly as the number of regions increases. The precision increases because each region captures more detailed information, i.e., user interests and language models. The recall decreases because it is more likely to classify a tweet to non-POI-related because of the increasing number of specific regions.

On the NYC dataset, both precision and recall are poor when the regions are fewer

than 30. One possible reason is that the POI-related tweets are distributed in many regions. When the number of regions is small, the regions learnt are too large to capture local information. When there are more than 30 regions, the model gets stable.

Based on the observations made from Figure 5.3, we set the number of regions at 30 on both datasets.

Figure 5.3: Effects of the number of Latent Regions (fixing the number of dummy POIs to be 10,000)

5.4.3 Overall Performance

We now compare the overall performance of the 9 methods listed in Table 5.3. For **sBM**, all the parameters are set as described in Section 3.4.1. The number of regions in **CLASS**-

R is set to the same number as the supervised Bayesian model. We train **sBM** on a single machine with Intel Xeon E5-1620 CPU and 16 GB RAM and it takes 21 minutes and 42 minutes for training on NYC and SG datasets, respectively. We run the 9 methods on the test set and evaluate the results using precision, recall, and F1 score. The performance of all methods are reported in Figure 5.4.

Figure 5.4: Precision, Recall, and F1 of all the 9 methods, on the two datasets NYC and SG

As shown in Figure 5.4(c), **sBM-T** and **sBM** significantly outperforms the other models. Specifically, **sBM** improves the F1 score of the second best solution **CLASS-LR** by 28.3% on the NYC data and outperforms the second best solution **RANK** by 20.4% on the SG data. Moreover, **sBM** consistently performs better than **sBM-T** on both datasets because it considers tips as external information.

RANK achieves considerable precision on the NYC dataset because the distance threshold reduces the search space. However, the recall is low because the threshold also excludes the correct POI in many cases. **RANK** achieves considerable recall but low precision on the SG dataset as the POIs are much denser in Singapore. This result suggests that a simple language model cannot distinguish the similar POIs within the search space.

CLASS performs worse than **RANK** in most cases because the distance model in **CLASS** is less effective in eliminating the POIs that are not related to the tweet. **CLASS-R** performs better on the SG dataset but worse on the NYC data. It has low recall because the regions in New York city are more diverse than those in Singapore, and the regions learnt by simple clustering algorithm do not well fit the POI-relatedness problem.

CLASS-W4 and **CLASS-R-W4** perform worst because 1) the W4 model does not model the relation between word and POI; and 2) the number of POI-related tweets takes a small portion of all tweets in our problem. The W4 model designed for POI-related tweets cannot leverage all the tweets to learn the user mobilities.

CLASS-LR and **CLASS-R-LR** both outperform their counterparts i.e., **CLASS** and **CLASS-R**, respectively. **CLASS-R-LR** achieves even higher precision than **sBM**. One possible reason is that it uses more features to identify the correct POI for a tweet. The region features in **CLASS-R** and **CLASS-R-LR** are helpful to increase the precision, but result in lower recall compared to **CLASS** and **CLASS-LR**. This is because the region features are used to filter more POIs that may not be related to the tweet.

Our proposed supervised Bayesian model captures the relations between relatedness response and other variables, and user interests on regions. Thus, the model is more likely to infer the correct POI for POI-related tweets, and thus can achieve higher accuracy.

5.4.4 Performance of POI-Relatedness Problem

Because the learning-to-rank baselines use the same classifier as **CLASS** and **CLASS-R** for the POI-relatedness problem, we compare our model **sBM** with the other three methods **RANK**, **CLASS**, and **CLASS-R**. The results of the POI-relatedness problem are evaluated by R-Precision, R-Recall and R-F1, see Figure 5.5.

Observe from Figure 5.5, **sBM** outperforms **CLASS** by 29.1% on the NYC dataset and outperforms **CLASS-R** by 6% on the SG dataset. As discussed in the previous subsection, the **CLASS-R** performs worse on the NYC dataset because of the simple clustering algorithm used. Compared to **CLASS**, the regions even bring adverse impact on its performance. However, the method works well on the SG data because POI-related tweets in Singapore is relatively densely distributed.

Benefiting from the relatedness response and dummy POIs, **sBM** better fits latent regions for the relatedness problem. Because the relatedness response drives the distribution of POI-related tweets to be identical in different regions, i.e., some regions could

be more likely to have POI-related tweets while others tend to include tweets that are not related to POIs.

(a) R-Precision

(b) R-Recall

(c) R-F1

Figure 5.5: R-Precision, R-Recall, and R-F1 of the four methods for the POI-Relatedness problem, on the two datasets

5.4.5 Empirical Study on Latent Regions

In this subsection, we compare the regions learnt by sBM and those learnt by k-means algorithm to demonstrate the effect of the relatedness response on regions. Specifically, we randomly pick two of the 30 regions learnt by our sBM model in each dataset. One of them has positive weight ω_{2+r} in the regression, while the other one has negative weight. Note that, positive weight indicates that the tweets posted in the region are more likely

(a) sBM - New York City

(b) k-means - New York City

(c) sBM - Singapore

(d) k-means - Singapore

Figure 5.6: Regions in both Datasets (sBM v.s. k-means)

to be POI-related. For convenience, we call the former as *positive region* and the latter as *negative region*. To draw the boundary of a region on a map, we compute the contour line of its bivariate Gaussian distribution at confidence level 0.95. For k-means algorithm, we first assign each training tweet a cluster. Then, for each cluster, we use its containing tweets to estimate the bivariate Gaussian distribution for it. We select the two regions that are closest to the two regions learnt by sBM, respectively. The regions learnt by the two methods and the tweets in the two datasets are plotted in Figure 5.6.

In Figure 5.6, blue points indicate the non-POI-related tweets, and red points indicate the POI-related tweets. Positive regions are plotted in green color, while negative regions

are plotted in yellow. In the example of New York city, **sBM** learns tighter positive region and the positive region has more POI-related tweets. On the other hand, **sBM** learns a negative region with fewer POI-related tweets. Similar observations are made on Singapore data. This is because the relatedness response affects the learning of latent regions. It learns regions that are fit to the POI relatedness problem.

5.5 Summary

We study the problem of annotating POIs with geo-tagged tweets. By exploring the text, coordinates and the user behaviors, we start with two basic solutions and then propose a supervised Bayesian model. We further extend the model by integrating external text sources of POIs, i.e., Foursquare tips, and show by experiment that our proposed supervised Bayesian model is effective for associating POI with geo-tagged tweets. The problem and solutions proposed in this chapter benefit many applications, including user behaviors analysis, POI recommendations, geo-textual data stream publish/subscription, to name a few.

Chapter 6

Mining Topical Aspects and Regions of Individuals

Recently, many efforts [YCM⁺13, YZS⁺15, WYC⁺15, YCZ⁺15] have been made to mining topics of individuals from geo-textual data for POI recommendation, location prediction, etc. The existing studies focus on mining general topics (e.g., entertainment, work) while ignoring the topical aspects (e.g., environment, service) which are important in recommendation. In addition, they ignore the user’s sentiment on topics expressed in the text. In this research problem, we aim to mine the topical aspects, topical regions and user’s interests in a unified way. We propose a unified probabilistic model to solve the problem and an efficient online recommendation algorithm based on the proposed model. Different from the existing methods, our approach takes account of the interdependency among aspect, region, category and sentiment. Modeling the interdependency in a unified model has the following benefits — (1) The learning of topical aspects and topical regions may benefit each other. For example, some words in the text are related to functional and spatial information of the region, e.g., “restaurant”, “New York”, etc. These words are more likely to appear in topical regions but less likely to appear in topical aspects; (2) We can apply the model to many applications, such as POI recommendation, user recommendation and aspect satisfaction analysis in regions. Experiments on real-world datasets show that the proposed model is effective in POI recommendation and user recommendation. We also demonstrate that our model can explain the reason of the recommendations.

In the following sections, we first define the problem of mining topical aspects and regions of individuals with consideration of sentiment in Section 6.1. Then, we introduce the proposed unified model in Section 6.2, and apply the model in POI recommendations, user recommendation, and aspect satisfaction analysis in regions in Section 6.3. Finally, we conduct extensive experiments in Section 6.4 to show the effectiveness of the proposed model and conclude the chapter in Section 6.5.

6.1 Problem Definition

Suppose we have a collection of historical user reviews on POIs, denoted by $D = \{d_1, d_2, \dots, d_{|D|}\}$. Each review is represented by a triple $\langle u_d, \mathbf{w}_d, l_d \rangle$, where u_d and $\mathbf{w}_d$ are the user and the set of words of d , respectively, and l_d represents the POI where review was posted. Each POI l is represented as a pair $\langle \widetilde{\mathbf{cd}}_l, \mathbf{c}_l \rangle$, where $\widetilde{\mathbf{cd}}_l$, $\mathbf{c}_l$ are the coordinates and the categories of the POI. The unified user interest model aims to find A aspects and R regions and the user's interests on aspects, regions and categories.

6.2 Sentiment-Aspect-Region Model

6.2.1 Intuitions

We aim at developing a model that is able to 1) extract latent variables, i.e., topical-aspect, sentiment, and topical-region from the review data; 2) capture the interdependencies among category, POI, user, words and the three latent variables; and 3) discover user's topical-region and topical-aspect preferences. To achieve these objectives, we exploit the following intuitions in designing our model:

Intuition 1: A user visits POIs in a topical-region because the region is geographically convenient to the user (e.g., close to her activity areas) and its topics (e.g., shopping street, education area, etc.) satisfy the user's interest. Each user has her own preferences on the topical-regions.

Intuition 2: A user rates highly of a POI because she likes some aspects of the POI. Such preferences might be indicated in her review. Some users like to check the price range of a restaurant first while others might be more concerned with the environment. Moreover, POIs in different categories may have different aspects of interest.

Intuition 3: A user decides to visit a POI in a region by considering the category, category-aware topical-aspects of the POI and the distance to it. For example, users may visit POIs of the restaurant category with good environment, but she may first consider the restaurants nearby.

Intuition 4: When a user writes a review on a POI, she will use words for both the aspects of the POI and her sentiments about the aspects. The user may also use words for the topical-region of the POI. For example, a review on a shop in Times Square may say: "This shop offers best prices in Times Square." The reviewer uses "price" for *aspect*, "best" for *sentiment* and "Times Square" for *region* to construct the review.

6.2.1.1 Model Description

Following the intuitions discussed in Section 6.2.1, we proceed to present our model. The notations used are listed in Table 6.1.

Table 6.1: Description of symbols

Symbol	Description
u, U	individual user and the set of users
l, L	individual POI and the set of POIs
c	category
r	topical-region
a, s	topical-aspect and sentiment
d, D	single review and the set of reviews
M	the number of sentences in a review
w, N	single word and the number of words in a sentence

Based on **Intuitions 1&2**, we model the user topical-region preferences and topical-aspect preferences as multinomial distributions $p(r|u)$ and $p(a|u, c)$, respectively.

Based on **Intuition 3**, a user chooses a POI to visit by considering both the category and the distance. We define the probability of visiting a POI l given category c and region r proportional to $p(l|c) \cdot p(l|r)$. Here $p(l|c)$ is the probability of selecting POI l from the category c ; $p(l|r)$ is a the probability of selecting POI l in region r by considering the distance from l to r . After normalization, we have the definition $p(l|c, r) = \frac{p(l|c)p(l|r)}{\sum_{l'} p(l'|c)p(l'|r)}$. To model the spatial distance, we use a Gaussian mixture model, i.e., $p(l|r) \sim N(\boldsymbol{\mu}_r, \boldsymbol{\Sigma}_r)$, where $\boldsymbol{\mu}_r$ is the center of region r and $\boldsymbol{\Sigma}_r$ is the co-variance matrix which depicts the area of region r . To model the membership of a POI to a category, we use a uniform distribution for $p(l|c)$.

Based on **Intuition 4**, we model the relationships among words, topical-aspects, sentiments and topical-regions by $p(w|a, s, r) = \lambda p(w|a, s) + (1 - \lambda)p(w|r)$, where a, s, r are topical-aspect, sentiment and topical-region, respectively. Here $p(w|a, s)$ is the probability that the users write word w when they have sentiment s on aspect a ; $p(w|r)$ is the probability that the users use word w to describe region r ; parameter λ is used to balance the portion of words drawn from topical-aspect, sentiment or topical-region. We model $p(w|a, s)$ instead of $p(w|a)$ and $p(w|s)$ because aspects and sentiments are closely coupled, and modeling by $p(w|a)$ and $p(w|s)$ needs an additional tuning parameter. Similar to proposals [TM08a, TM08b, JO11], we assume each sentence expresses opinions on exactly one topical-aspect and each topical-aspect is associated to a positive, negative or neutral sentiment.

In summary, the graphical representation of our model is shown in Figure 6.1 and the generative process of the reviews written by user u is described as follows:

- For each review $d \in D_u$, where D_u is the set of reviews written by user u .
 - Draw topical region $r \sim p(r|u)$

Figure 6.1: Sentiment-Aspect-Region model (SAR)

- Draw category $c \sim p(c|u)$
- Draw location $l \sim p(l|c, r) = \frac{p(l|r)p(l|c)}{\sum_{l'} p(l'|c)p(l'|r)}$, where $p(l|r) \sim N(\boldsymbol{\mu}_r, \boldsymbol{\Sigma}_r)$
- For each sentence in review d
 - * Draw aspect $a \sim p(a|u, c)$
 - * Draw sentiment $s \sim p(s|a, l)$
 - * For each word position in the sentence
 - Draw word $w \sim p(w|a, s, r) = \lambda p(w|a, s) + (1 - \lambda)p(w|r)$

In the model, $p(l|c)$ and $p(c|u)$ can be estimated directly from a given corpus. The other distribution parameters need to be inferred. We first present how to estimate $p(l|c)$ and $p(c|u)$, and then show the learning algorithm for the remaining distributions in Section 6.2.2.

As described in **Intuition 3**, a POI l is generated from both category and region. Since POI l and category c are observable variables, we simply compute $p(l|c)$ as shown in Equation (6.1).

$$p(l|c) = \frac{I(l, c)}{\# \text{ of POIs in } c} \quad (6.1)$$

$$I(l, c) = \begin{cases} 1 & l \in c \\ 0 & \text{otherwise} \end{cases} \quad (6.2)$$

Similarly, we compute the category preferences of each user, i.e., $p(c|u)$, directly from the corpus. To handle the overfitting problem, we apply the additive smoothing technique. After smoothing, even though a user did not visit some category of POIs, she will still visit that category with a low probability. The computation of $p(c|u)$ is shown in Equation (6.3).

$$p(c|u) = \frac{n_c + \alpha}{N + \alpha C}, \quad (6.3)$$

where n_c is the number of reviews of POIs in category c that a user u writes; N is the total number of reviews on POIs in c ; C is the total number of categories; α is the smoothing parameter which is usually set to a value smaller than 1. Empirically, α is set at 0.1.

6.2.2 Learning Algorithm

To learn the parameters of the model, we use the expectation-maximization (EM) approach. In this section, we present the computation of the corpus likelihood, the two-step EM algorithm used to learn our parameters, and initialization of the EM algorithm.

Likelihood Computation. Our model has several levels, i.e., word level, sentence level, and document level. The latent variables are on two levels. Region r is at document level while aspect a and sentiment s are at sentence level. This multi-level structure poses challenges to the estimation of the log-likelihood. According to the generative process, we have the likelihood of the corpus D :

$$p(D; \Phi) = \prod_d^D p(u_d) \sum_r^R p(r|u_d) p(l_d, \mathbf{w}_d | r, u_d) \quad (6.4)$$

$$p(l_d, \mathbf{w}_d | r, u_d) = p(c_{l_d} | u_d) p(l_d | r, c_{l_d}) \prod_i^M p(\mathbf{w}_{d,i} | c_{l_d}, r, u_d, l_d) \quad (6.5)$$

$$\begin{aligned} & p(\mathbf{w}_{d,i} | c_{l_d}, r, u_d, l_d) \\ &= \sum_{a,s} p(a | c_{l_d}, u_d) p(s | a, l_d) \prod_j^N p(w_{d,i,j} | a, s, r) \end{aligned} \quad (6.6)$$

In Equation (6.4), Φ is the set of parameters in the model, i.e., $p(r|u)$, $p(a|c, u)$, $p(l|r)$, $p(s|a, l)$, $p(w|a, s)$, $p(w|r)$, $\boldsymbol{\mu}_r$ and $\boldsymbol{\Sigma}_r$. Variables $u_d, l_d, \mathbf{w}_d$ are the user, location and words of review d , respectively. Variable $\mathbf{w}_{d,i}$ represents the set of words in sentence i of review d while $w_{d,i,j}$ is the j^{th} word in sentence i of review d . Taking logarithm of $p(D; \Phi)$ leads to a summation inside the logarithm:

$$L = \sum_d \log p(u_d) + \log \sum_r p(r|u_d) p(l_d, \mathbf{w}_d | r, u_d) \quad (6.7)$$

Since this likelihood cannot be estimated directly, we adopt Jensen's inequality to the log-likelihood, and estimate the lower bound of the likelihood and the parameters in an iterative manner.

Expectation-Maximization. Due to the aforementioned difficulty of computing log-likelihood directly, we apply Expectation-Maximization (EM) algorithm to estimate the model parameters.

In **E-step**, we compute the expectation of latent variables given the observed data. By applying Jensen's inequality to Equation (6.7), we get the lower bound of the likelihood as:

$$L_{LB} = \sum_d \log p(u_d) + \sum_{d,r} p(r|d) (\log p(r|u_d) + \log p(l_d, \mathbf{w}_d|r, u_d)) \quad (6.8)$$

As shown in Equation (6.8), we need to estimate $p(r|d)$ to compute the full likelihood. We apply Bayes rule, and obtain the update function of the posterior distribution as

$$p(r|d) = \frac{p(r, d)}{\sum_r p(r, d)} \quad (6.9)$$

$$p(r, d) = p(u_d)p(r|u_d)p(l_d, \mathbf{w}_d|r, u_d) \quad (6.10)$$

In Equation (6.10), $p(l_d, \mathbf{w}_d|r, u_d)$ is computed by Equation (6.5), and $p(u_d)$ appears both in the numerator and the denominator, and thus is not necessary to estimate.

In **M-step**, by maximizing the lower bound of likelihood, we can obtain the update function of parameters at document level that are related to topical region r as below.

$$p(r|u) = \frac{\sum_{d \in D_u} p(r|d)}{\sum_r \sum_{d \in D_u} p(r|d)} \quad (6.11)$$

However, we cannot obtain a close form solution for $\boldsymbol{\mu}_r$ and $\boldsymbol{\Sigma}_r$ due to the normalization term. We adopt a gradient method to obtain the update value of $\boldsymbol{\mu}_r$ and $\boldsymbol{\Sigma}_r$ in M-step. Specifically, we use the BFGS quasi-Newton method [KIH⁺13, LN89]. In the gradient method, we compute the gradient of $\boldsymbol{\mu}_r$ and $\boldsymbol{\Sigma}_r$ as follows:

$$\frac{\partial L_{LB}}{\partial \boldsymbol{\mu}_r} = \sum_d p(r|d) \boldsymbol{\Sigma}_r^{-1} \left(\frac{\sum_{l'} q(l') (\mathbf{c} \mathbf{d}_{l'} - \boldsymbol{\mu}_r)}{\sum_{l'} q(l')} - (\mathbf{c} \mathbf{d}_{l_d} - \boldsymbol{\mu}_r) \right) \quad (6.12)$$

$$\frac{\partial L_{LB}}{\partial \boldsymbol{\Sigma}_r} = \sum_d p(r|d) \left(\frac{\sum_{l'} q(l') g(l', r)}{\sum_{l'} q(l')} - g(l_d, r) \right), \quad (6.13)$$

where $q(l') = p(l'|c_l)p(l'|r)$ and $\mathbf{cd}_l$ denotes the coordinates of POI l . The function $g(l, r)$ in Equation (6.13) is the gradient of the Gaussian distribution for region r w.r.t. Σ_r at point l .

Since sentiment and aspect are at the sentence level, we cannot compute $\log p(l_d, \mathbf{w}_d|r, u_d)$ in Equation (6.8) using $p(r|d)$. We introduce a new latent variable to estimate parameters related to aspect and sentiment. Specifically, we use ϕ_{a,s,r,d_i} to identify the probability that the i^{th} sentence in a review d from region r is assigned with aspect a and sentiment s . we use ϕ_{a,s,r,d_i} and $p(r|d)$ to compute the update function of $p(a|c, u)$, $p(s|l, a)$, $p(w|a, s)$, and $p(w|r)$.

Denote by $n(w, d_i)$ the number of occurrences of word w in sentence i of review d . We estimate ϕ_{a,s,r,d_i} as:

$$\phi_{a,s,r,d_i} = \frac{p(a, s, r, d_i)}{\sum_{a,s} p(a, s, r, d_i)} \quad (6.14)$$

$$\begin{aligned} p(a, s, r, d_i) &= p(u_d)p(r|u_d)p(c_{l_d}|u_d, r)p(l_d|r, c_{l_d}) \\ &\quad p(a|c_{l_d}, u_d)p(s|a, l_d) \prod_w p(w|a, s, r)^{n(w, d_i)} \end{aligned} \quad (6.15)$$

By maximizing the lower bound of the likelihood, we obtain the update function of the rest parameters:

$$p(a|u, c) = \frac{\sum_{d \in D_u} \sum_r p(r|d) \sum_i \sum_s \phi_{a,s,r,d_i}}{\sum_{a'} \sum_{d \in D_u} \sum_r p(r|d) \sum_i \sum_s \phi_{a',s,r,d_i}} \quad (6.16)$$

$$p(s|l, a) = \frac{\sum_{d \in D_l} \sum_r p(r|d) \sum_i \sum_s \phi_{a,s,r,d_i}}{\sum_{s'} \sum_{d \in D_l} \sum_r p(r|d) \sum_i \sum_s \phi_{a,s',r,d_i}} \quad (6.17)$$

$$p(w|s, a) = \frac{\sum_d \sum_r p(r|d) \sum_i \phi_{a,s,r,d_i} n(w, d_i)}{\sum_{w'} \sum_d \sum_r p(r|d) \sum_i \phi_{a,s,r,d_i} n(w', d_i)} \quad (6.18)$$

$$p(w|r) = \frac{\sum_d p(r|d) \sum_i \sum_a \sum_s \phi_{a,s,r,d_i} n(w, d_i)}{\sum_{w'} \sum_d p(r|d) \sum_i \sum_a \sum_s \phi_{a,s,r,d_i} n(w', d_i)}, \quad (6.19)$$

where D_u is the set of reviews written by user u and D_l is the set of reviews for POI l .

Initialization of EM Algorithm. EM algorithm can only guarantee to find a local optima. Different initializations may lead to different results. In this section, we present our methods for initializing the assignment of aspect, sentiment and region.

Aspect is extracted from sentence level in our model. We initialize the aspect by a clustering process on sentences. Each sentence is represented as a vector of words. Given the number of aspects, we use K-means clustering algorithm to assign each sentence an aspect. We then initialize $p(w|a)$ by the probability that word w appears in sentences carrying aspect a .

Sentiment has 3 possible values: positive, negative and neutral. In order to know the polarity of each sentiment, we need some prior knowledge. We use the same predefined set of sentiment seed words as in Jo’s proposal [JO11]. Moreover, we apply a syntactic parser to extract negation of the sentiment words such as “not good” and use a special word “not_good” to represent the phrase “not good” in our vocabulary. For each word in the seed word set, we assign a probability ($p(w|s)$) of 1 to its polarity and 0 to the other two polarities. For words not in the seed word set, we assign an equal probability for each polarity. We then use $p(w|a)p(w|s)$ to approximate $p(w|a, s)$.

Region is initialized by a K-means clustering algorithm based on the coordinates (latitude and longitude). The clustering algorithm partitions POIs to different regions. Then for each region r , we compute μ_r and Σ_r using a regression over the POIs in the region. We compute $p(w|r)$ by the distribution of words in the reviews for POIs in region r and $p(r|u)$ by the portion of reviews that user u writes in region r .

For other parameters: $p(a|c, u)$ and $p(s|a, l)$, we initialize them by using the assignment of aspect and sentiment to a sentence (We assign sentiment to a sentence by voting from sentiment seed words extracted from the sentence). Specifically, $p(a|c, u)$ is proportional to the number of sentences that are assigned to a and that belong to a review written by u from category c ; $p(s|a, l)$ is proportional to the number of sentences that belong to location l and are assigned to sentiment s and aspect a at the same time.

6.2.3 Efficiency Analysis

Let the number of sentiment be 3 and we treat it as constant. In E-step, the computation of the expectation of latent variables in Equation (6.9) and the variables ϕ_{a,s,r,d_i} in Equation (6.19) needs $O(|D|MNRA) = O(WRA)$, where W is the number of words in the reviews of all users’ in training set, R is the number of regions and A is the number of aspects. In M-step, the cost for updating Equation (6.16) to (6.19) is $O(UA + LA + VA + VR)$, where U, L, V are the number of users, POIs and unique words, respectively. To update μ and Σ , we perform a quasi-Newton method. Since each μ_r and Σ_r are two dimensional vector and 2×2 matrix, respectively. The computation cost of matrix operation can be treated as constant. Let D be the number of reviews, the cost of computing gradient in Equation (6.12) and (6.13) is $D + L$. Therefore, the complexity of quasi-Newton is $O(I_q R(D + L))$, where I_q is the number of iterations of quasi-Newton. In summary, the total complexity of the learning algorithm with I iterations is $O(I(WRA + I_q R(D + L) + UA + LA + VA + VR))$. Since $WRA \gg (UA + LA + VA + VR)$, we simplify the cost as $O(I(WRA + I_q R(D + L)))$. We can parallelize the computation of both E-step and M-step. In E-step, since the computation of $p(r|d)$ on each document is independent to others, we can compute $p(r|d)$ of each document in parallel. In M-step, the update of Equation (6.16) to (6.19) and the quasi-Newton iterations can also be parallelized in the similar way as $p(r|d)$. Therefore, the algorithm can be fully parallelized.

6.3 Applications

We present three applications of our model, namely POI recommendations, user recommendations, and aspect satisfaction in regions. In POI recommendation, we provide a way to explain the reason of recommending a POI and propose an efficient online recommendation algorithm.

6.3.1 POI recommendation

We apply our model to two POI recommendation tasks and propose an efficient online recommendation algorithm. The two recommendation tasks are *All-Category Recommendation* and *Single-Category Recommendation*.

6.3.1.1 All-Category Recommendation

All-Category Recommendation is a task of generating a rank list of POIs in any category given a set of POIs and a user. The aforementioned proposals are all for all-category recommendation. We calculate the probability of $p(l, s_+|u)$, i.e., the probability of user u visits POI l with positive sentiment, to score l for u as shown in Equation (6.20).

$$p(l, s_+|u) = \sum_r p(r|u)p(c_l|u)p(l|r, c_l) \sum_a p(a|u, c_l)p(s_+|a, l) \quad (6.20)$$

According to Equation (6.20), we make the recommendation by considering the matching between user preferences (i.e., $p(r|u)$, $p(c_l|u)$ and $p(a|u, c_l)$) and the attributes of the POI (i.e., $p(s_+|a, l)$ and $p(l|r, c_l)$).

This recommendation model enables us to explain why we recommend a POI to a user. We consider two factors: aspect and region. First, we rank the aspects by $p(s_+|a, l)p(a|u, c_l)$ to reveal which aspects match the user's preferences. Second, we rank the regions by $p(r|u)p(l|r)$ to reveal which regions contribute more to the recommendation. Finally, we choose top several aspects and regions for explanation.

6.3.1.2 Single-Category Recommendation

Single-Category Recommendation aims at ranking POIs given a user and a specific category (e.g., restaurants). It is a typical scenario for POI recommendation although it has not been covered in previous work. We compute $p(l, s_+|u, c)$ as shown in Equation (6.21).

Compared to all-category recommendation, we fix the category i.e., remove $p(c|u)$ from Equation (6.20). All locations that are not in c will not be considered in this scenario.

$$p(l, s_+|u, c) = \sum_r p(r|u)p(l|r, c) \sum_a p(a|u, c)p(s_+|a, l) \quad (6.21)$$

We can also offer explanation for the single-category recommendation by following similar method as we employ for the all-category recommendation.

6.3.1.3 Efficient algorithm for Top-N Online Recommendation

Time efficiency is an essential part of online recommendation. A straightforward method of making recommendation is to compute the recommendation score as Equation (6.20) or Equation (6.21). This method requires traversing all the regions which is highly time consuming. The threshold algorithm [FLN01] can be applied to reduce the number of POIs considered and thus reducing the computation time. However, in our applications, the number of regions and aspects is large, and thus it is expensive to compute the recommendation score even for a single POI. The threshold algorithm cannot help with this, either. Due to the aforementioned reasons, we propose an optimized top-N items recommendation algorithm that significantly reduces the time cost. As to be shown in the experiment, our algorithm is faster than the threshold algorithm in the top-N POI recommendation using our model. Our algorithm can be applied to all or single-category POI recommendation. We use all-category POI recommendation (Equation (6.20)) as an example to explain the algorithm.

Our algorithm is designed according to the following observations: 1) A user often prefers to visit a small number of regions; and 2) POIs in the center of the region are more likely to be recommended. These two observations indicate that only when a user prefers a region and the POI is near the center of the region, will the score $p(r|u)p(l|r, c_l)$ contribute significantly to the recommendation score. Therefore, after we have computed the most possible regions for a POI, it may not be necessary to compute the remaining regions. Specifically, we design a branch and bound algorithm to prune the search space of the regions. Algorithm 5 shows the details of our algorithm.

Our algorithm takes two steps — (1) *initialization* and (2) *pruning*. In the *initialization* step (line 1), we find N candidate POIs that are potentially good for recommendation. Specifically, we pick top K regions which cover most of the user's regional preferences (i.e., $\sum_{i=1}^K p(r_i|u) > 0.9$) with smallest K (line 19). If K is larger than N , we pick at most N regions to ensure that we can select at least one candidate from each region. In each of the top K region, we choose top $\lceil \frac{N}{K} \rceil$ POIs w.r.t. $p(l|r)$ as candidates.

Algorithm 5: POI Recommendation

Input: A user u and a number N
Output: The top N recommended POIs to u
begin

```

1   $H \leftarrow \text{InitialCandidates}(N);$ 
2  for  $l \in L$  and  $l \notin H$  do
3       $PartS \leftarrow 0, PartRPro \leftarrow 0, Skip \leftarrow false;$ 
4      while there exists  $r$  not examined for  $l$  do
5           $r \leftarrow \text{next region};$ 
6           $PartS \leftarrow PartS + p(r|u)p(l|r, c_l);$ 
7           $PartRPro \leftarrow PartRPro + p(r|u);$ 
8          if  $PartS + (1 - PartRPro) * p(l|r, c_l) < H.Top()$  then
9               $Skip \leftarrow true, break;$ 
10         if  $Skip = false$  then
11              $S \leftarrow PartS * p(c_l|u) \sum_a p(s_+|a, l)p(a|u, c_l);$ 
12             if  $S > H.Top()$  then
13                  $H.DeleteTop();$ 
14                  $H.Insert(< l, S >);$ 
15  $Result \leftarrow \text{Sort } H \text{ by Score } S;$ 
16 return  $Result;$ 
```

Function InitialCandidates(N)

```

17  $H \leftarrow \emptyset;$ 
18  $r_1, \dots, r_R \leftarrow \text{Sort the regions by } p(r|u);$ 
19 Pick top  $K$  regions satisfies:  $K = \min(\{k | \sum_{i=1}^k p(r_i|u) > 0.9\}, N);$ 
20 From  $r_1$  to  $r_K$ , Insert top  $\lceil \frac{N}{K} \rceil$  POIs ordered by  $p(l|r)$  to  $H$  until  $H$  contains
     $N$  POIs;
21 return  $H;$ 
```

In the *pruning* step (line 8-9), we check whether we can avoid traversing unnecessary regions for each POI. We traverse the regions according to the descending order of $p(l|r, c_l)$ for POI l . Suppose we have traversed regions $\{r_1, \dots, r_{i-1}\}$. The partial recommendation score accumulated by the traversed regions is

$$PScore = \sum_{j=1}^{i-1} p(r_j|u)p(l|r_j, c_l).$$

When we integrate the i -th region, the upper bound of recommendation score for the

POI is computed as:

$$Bound^{(i)}(l) = PScore + (1 - \sum_{j=1}^{i-1} p(r_j|u))p(l|r_i, c_l). \quad (6.22)$$

As the regions are traversed in the descending order of $p(l|r, c_l)$, the actual value of $p(l|r, c_l)$ for the remaining regions should be less than the one for the current region, i.e., $p(l|r_i, c_l)$. Therefore, we have a partial recommendation score for the rest of the regions, which is at most

$$(1 - \sum_{j=1}^{i-1} p(r_j|u))p(l|r_i, c_l),$$

where $1 - \sum_{j=1}^{i-1} p(r_j|u)$ is the user interests on the remaining regions. The upper bound of $\sum_r p(l|r, c) p(r|u)$ in Equation (6.20) is $PScore + (1 - \sum_{r=r_1}^{r_{i-1}} p(r|u))p(l|r_i, c_l)$. Since $\sum_a p(a|u, c) p(s_+|a, l) \leq 1$, we obtain the upper bound of the recommendation score in Equation (6.20) for the POI by setting $\sum_a p(a|u, c) p(s_+|a, l) = 1$, which results in Equation (6.22).

If the upper bound is smaller than the N^{th} candidate (Line 8), we skip the current POI without checking the remaining regions. Otherwise, we continue to traverse the remaining regions to compute the recommendation score of the POI. If all regions are traversed and the POI is not pruned by the aforementioned upper bound, we compute the full score, i.e., Equation (6.20), and compare with the N^{th} smallest candidate (line 11). If the recommendation score of the current POI is larger than the N^{th} candidate, we replace the N^{th} candidate with the current POI. To maintain the top-N candidate list, we use a binary min-heap.

6.3.2 User Recommendation

We can also apply our model to recommend users for a POI. Predicting which users may favor a given POI is useful when the owner of the POI wants to target at or advertise to some of the users. Given a POI l , we compute the probability $p(u, s_+|l)$ of user u favoring POI l by considering both topical-region and topical-aspect preferences of users as follows:

$$p(u, s_+|l) = \frac{p(u, s_+, l)}{\sum_{u, s} p(u, s, l)} \quad (6.23)$$

$$\begin{aligned} p(u, s, l) = & p(u) p(c_l|u) \sum_r p(r|u) p(l|r, c_l) \\ & \sum_a p(a|u, c_l) p(s|a, l), \end{aligned} \quad (6.24)$$

where prior $p(u)$ is calculated using the user's review history:

$$p(u) = \frac{\# \text{ of reviews } u \text{ wrote}}{\# \text{ of all reviews}}.$$

Since the last two summations are the same as those in POI recommendation, Algorithm 5 can also be used to speed up the user recommendation.

6.3.3 Aspect Satisfaction in Regions

Discovering which aspect is satisfied or dissatisfied by users in each region is useful when 1) someone wants to set up a new business or make strategies to attract more customers, or 2) policy makers make urban planning. For example, most of the restaurants in a region of a city may be complained for the long waiting time. By knowing the dissatisfaction of this aspect, a restaurant may think how to achieve competitive advantage over other restaurants in the region. We can infer the aspect satisfaction in each region based on our model. Specifically, we compute the aspect distribution of each sentiment s , category c and region r as

$$p(a|s, c, r) = \frac{\sum_{u,l} p(u)p(r|u)p(c|u)p(a|c, u)p(l|r, c)p(s|a, l)}{\sum_{a,u,l} p(u)p(r|u)p(c|u)p(a|c, u)p(l|r, c)p(s|a, l)} \quad (6.25)$$

This probability shows which aspect is most probably liked/disliked in POIs from category c and region r .

6.4 Experiments

We conduct several experiments to evaluate the performance of our model by comparing with the state-of-the-art techniques. First, we present the experimental setup in Section 6.4.1 and the model selection strategy in Section 6.4.2; Then, we apply our model to the three applications. We show the accuracy and efficiency of our recommendation algorithm in POI recommendation in Section 6.4.3 and explore the reason of making a POI recommendation in Section 6.4.4. After that, we show the accuracy of user recommendation in Section 6.4.5. Finally, we analyze the aspect satisfaction in regions in Section 6.4.6 and discuss the quality of topical regions in Section 6.4.7.

6.4.1 Experimental Setup

Although we have associated POIs with geo-tagged tweets in Chapter 5, such data does not contain the ground truth of whether a user really likes a POI. For evaluation purpose, we use Yelp dataset, which contains users' ratings in each geo-tagged review. Specifically,

we collect data from two different cities in Yelp, which hosts hundreds of thousands of user reviews and ratings for POIs. One of our datasets is from Yelp’s Challenge Dataset¹, which contains 11,537 POIs and 43,873 users from Phoenix at the time of data collection. We remove users who wrote fewer than 2 reviews and POIs without any reviews after filtering the users. This preprocessing results in a dataset containing 11,359 POIs and 21,908 users. We crawl the other dataset from Yelp Singapore, and remove users and POIs without any reviews. The resulting Singapore dataset contains 8,846 POIs and 1,654 users. For both datasets, we also remove the English stop words and highly frequent words that appear more than 10,000 times in the corpus. The statistics of the two datasets is shown in Table 6.2. For each dataset, we hold out the recent 10% reviews of each user for tuning and 10% for testing, respectively, and use the remaining 80% data as training set. In the tuning and test data, we generate the ground truth, i.e., whether a user likes the POI or not, by checking how the user rates to that POI. When the user rates more than 3.5 stars, we consider that she likes the POI and put the user-POI pair into the set of ground truth.

Table 6.2: Statistics of the two datasets

	Phoenix	Singapore
#POIs	11,359	8,846
#Users	21,908	1,654
#Reviews	215,837	20,248
#Reviews per user	4.92	12.40
#Reviews per POI	18.71	2.36

We run all experiments on a machine with Intel Xeon E5-2680 (2.8 Ghz) 10-cores CPU and 64GB memory. We deploy the training program on 8 cores of the CPU and train the SAR models on Singapore and Phoenix datasets with the same settings as in Section 6.4.2. The training time of Singapore data is 33.74 minutes and the Phoenix data is 965.61 minutes.

6.4.2 Model Selection

In our model, we have two free parameters: number of aspects A and number of topical regions R , and a tunable parameter λ . To find a proper value of A and R , we use Bayesian Information Criterion (BIC) which is usually used for model selection. BIC is defined as: $BIC = -2L + K\ln(N)$, where L is the likelihood of the model, K is the number of parameters and N is the scale of the corpus. In our scenario, K is $A \times R$, while N is the number of review sentences. We set the default value of parameter λ to 0.6, and choose

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

A, R with lowest BIC for each dataset. In the Phoenix data, the lowest BIC appears when $R = 80$ and $A = 30$. In the Singapore data, smaller number of aspects and regions are preferred because Singapore is smaller than Phoenix. We set $R = 40$ and $A = 20$ for Singapore data.

6.4.3 POI recommendation

First, we introduce several existing methods for competition and the evaluation metrics. Then, we compare with the state-of-the-art baselines for both all-category and single-category recommendation. Finally, we evaluate the efficiency of our recommendation algorithm.

6.4.3.1 Recommendation Methods

We compare our model to five POI recommendation techniques.

- **CF**: The basic user-based collaborative filtering model.
- **GCF**: A collaborative filtering model incorporating geographical influence [YYLL11].
- **W3**: A topic model considering topics in user-generated text and activity regions for individual user [YCM⁺13].
- **STM**: A topic model with global regions [HE13].
- **EFM**: A matrix factorization based explicit factor model which extracts aspects and sentiment from reviews models the relation among user, item, aspect and sentiment for recommendation [ZLZ⁺14].

6.4.3.2 Evaluation Metrics

Evaluating a recommended list has two ways: one of them is how many true results are hit by the list and the other is how similar the resulting rank and the ground truth rank are. Therefore, we use two kinds of metrics to measure the performance of our model and the peers. These metrics are: 1) the precision and recall for the top N items, namely *Precision@N* and *Recall@N*, respectively. We investigate $N = 5$ and $N = 10$ because the top few results are most impressive to users. 2) *Mean Average Precision* (MAP) which is used to show the correctness of a rank list according to the position of true results in the list. If the true results are ranked high in the list, the list is probably a good recommendation result.

6.4.3.3 All-Category POI Recommendation

The result is shown in Figure 6.2 and Figure 6.3. All results reported in this section pass t-test with $p\text{-value} < 0.01$, which means the improvements are significant. Our SAR model outperforms the best peer by 37% , 42% and 70% in terms of Precision@10, Recall@10 and MAP, respectively on the Phoenix dataset, while 28% , 37% and 44% in terms of Precision@10, Recall@10 and MAP, respectively on the Singapore dataset.

Among the baseline methods, CF and EFM do not consider geographical information, which limits the performance of these two methods. Compared to CF, EFM performs better because it explores the user preferences on aspect level. W3 model considers latent regions. However, it does not consider aspect and sentiment. STM performs worst because it estimates the probability $p(l|r)$ by the probability density function of Gaussian distribution without any normalization. The Gaussian distribution in STM overwhelms the other probabilities (i.e., the preferences of the user). GCF incorporates the graphical information into the model of CF. However, without considering the content of the reviews, GCF cannot reveal user's preferences on aspect level.

Our SAR model discovers user's latent interest on several factors: aspect, sentiment, category, and region. Benefiting from the user preference analysis on topical-aspects and topical-regions, SAR model outperforms these methods.

6.4.3.4 Single-Category POI recommendation

The peers are developed for all-category recommendations. To apply them to this task, we consider two methods. The first one is to pick the top N results that belong to the given category from the all-category recommendation results. The second one is to divide the visit history of each user by categories and learn the models on data from each category separately. The second method suffers from the problem of sparser data. Therefore, we adopt the first method in this experiment.

The result is reported in Figure 6.4 and Figure 6.5. Our SAR model outperforms the best peer by 40% , 41% and 46% in terms of Precision@10, Recall@10 and MAP, respectively on the Phoenix dataset, while 15% , 27% and 29% in terms of Precision@10, Recall@10 and MAP, respectively on the Singapore dataset. The reason is that our model is able to discover the relation between category and aspect by modeling the user preferences to topical-aspects on each category, i.e., $p(a|c, u)$.

6.4.3.5 Recommendation Efficiency

To evaluate the efficiency of the proposed branch and bound algorithm for POI recommendation (B&B), we run the algorithm on both of the datasets and compare the time consumption with two algorithms. One of them is the brute-force algorithm which computes Equation (6.20) and uses a partial sorting method to find the top-N result. The

(a) Pre@N - Phoenix

(b) Rec@N - Phoenix

(c) Pre@N - Singapore

(d) Rec@N - Singapore

Figure 6.2: Precision@N and Recall@N for All-category POI recommendation

other one is the threshold algorithm (TA) [FLN01] that sorts $p(l|r, c_l)$ for each region and accesses the POIs on each sorted list in parallel. Since the scoring function is a monotonic increasing function, we follow the threshold algorithm to find the top-N results. We do not sort by aspects over POIs because the differences among POIs on an aspect ($p(s|a, l)$) is much smaller than those on a region. Since $\sum_a p(a|u, c)p(s|a, l)$ and $p(c|u)$ in Equation (6.20) are no larger than 1, we compute the threshold as: $T = \sum_r p(r|u) \max_{l'} p(l'|r, c_{l'})$.

In this experiment, we retrieve top 100 results for each user. To investigate the running time on POIs of different scales, we randomly select subsets with different sizes from the two datasets. Specifically, we get 10 different subsets with size from 1000 to 10,000 POIs in Phoenix data while another 10 subsets with size from 800 to 8000 POIs

Figure 6.3: MAP for All-category POI recommendation

in Singapore data. The time of recommending top 100 POIs to a single user is computed by averaging over all users. The result is reported in Figure 6.6.

Our optimized algorithm, namely **B&B** in Figure 6.6, always takes the least amount of time on both datasets for different number of POIs. Compared to **TA**, we achieve 4 times faster in the Phoenix data and 2 times faster in the Singapore data. The reason could be that **TA** has to update the threshold for each access on any sorted list. This extra computation makes **TA** perform even worse than the brute-force algorithm when the number of POIs is small. We give a more detailed comparison for the threshold algorithm and our methods as below.

The threshold algorithm still need to access many of the POIs (approximately 40% of the POIs in Phoenix on average) in each sorted list for each region. However, our algorithm accesses only few regions to compute the partial score (less than 10 regions on average in Phoenix) for each POI, and we only compute the full score for POIs that we access all the regions (less than 10% of the POIs in Phoenix). Suppose there are L POIs, R regions and A aspects and we compute the full score for $0.4L$ POIs in threshold algorithm and $0.1L$ POIs in our algorithm². In the worst case, the threshold algorithm updates $0.4LR$ times of threshold. The total computation cost for threshold algorithm is $0.4L(A + 2R)$. Our algorithm needs to compute the full score for $0.1L$ POIs as well as the partial score and the threshold for the rest POIs on 10 regions on average. The computation costs is $0.1L(A + R) + 0.9L \times 2 \times 10$. When $A = 30$ and $R = 80$ in the Phoenix data, the cost of threshold algorithm is $76L$ while the cost of our algorithm is $29L$. When the number of regions increases, the threshold algorithm has more sorted list to access, which makes it slower. Whereas in our algorithm, we still need to consider only a few regions (near a POI). Our algorithm is more suitable for the models that have a large number of regions.

²We estimate this average ratio on our datasets

(a) Pre@N - Phoenix

(b) Rec@N - Phoenix

(c) Pre@N - Singapore

(d) Rec@N - Singapore

Figure 6.4: Precision@N and Recall@N for Single-category POI recommendation

6.4.4 Explanation of Recommendation

As discussed in Section 6.4.3, the SAR model can tell why we recommend or not recommend a POI. To illustrate this, we randomly pick some examples from the test data. We explore aspects and regions respectively. To find out which aspects contribute most to the recommendation, we first show the top-5 favorite aspects of the selected user in the category of the POI according to $p(a|u, c)$. Then we report the top-5 good aspects of the POI according to $p(s_+|a, l)$. We manually give a name to aspect according to the word distribution. Table 6.3 shows three users and a recommended POI for each of them. *User 64* prefers the food and flavor aspects of a restaurant and *Paradise Dynasty*, a restaurant of Chinese food, has positive reviews on foods. *User 121* wants a good environment in

Figure 6.5: MAP for Single-category POI recommendation

a bar, and *Wala Wala Cafe Bar* has a good environment. *User 420* prefer a hotel with good facility and service and *Marina Bay Sands* provides good facility although only has neutral reviews on service. This table shows that we recommend to users the POIs with aspects that match their preferences. This explains why our method makes a recommendation to a user. This is a very desirable feature for a recommendation system, although many existing recommendation methods cannot offer such explanation.

To explore the influence of regions, we draw the contour of the top 3 regions of *User 64* ranked by probability $p(r|u)$, and plot the top 5 recommended POIs to the user. We highlight *Paradise Dynasty* in blue color. Figure 6.7 shows the regions and recommended POIs. The 5 POIs are close to the center of *Region 36* or *Region 8*, two of the user's favorite regions, which is the geographical reason for recommending those POIs. Our recommendation algorithm tends to recommend POIs that are close to the center of the user's favorite regions.

6.4.5 User Recommendation

6.4.5.1 Methods and Metrics

W3 and STM, which are based on topic models, can be easily applied to user recommendation by multiplying the conditional probability $p(l|u)$ with user popularity $p(u)$. However, the existing CF (CF and GCF) and EFM baselines are not designed for user recommendation, we simply reverse the rating matrix by treating users as items and items as users in CF methods. For GCF model, we compute the score in the geographic preference matrix by comparing the coordinates of the POI with each user's visit histories, and normalizing over all users:

$$Score_{geo}(l, u) = \frac{\prod_{l' \in L_u} Dist(l, l')}{\sum_{u'} \prod_{l' \in L_{u'}} Dist(l, l')} \quad (6.26)$$

Figure 6.6: All-category POI recommendation time consumption

The set L_u is the visit history of user u . We then linearly combine the geographic preference score and the visit-based score as the original model does. For EFM, we rank users by $R_{i,j}$ (i.e., the predicted rating of user i on item j) for each item. We use the same metrics as those in Section 6.4.3 to measure the performance of user recommendation.

6.4.5.2 Result

Figure 6.8 and Figure 6.9 show the comparison among our model and the peers. The results reported are significant with $p\text{-value} < 0.01$ in t-test. The SAR model outperforms the peers because we model the interaction among aspect, sentiment and region. The interaction identifies whether a user likes a region (i.e., $p(r|u)$) and an aspect given a category (i.e., $p(a|u, c)$). When performing recommendation, we check whether the given POI satisfies the preferences of users. The SAR model does not perform significantly better than the other methods in Singapore because the number of reviews per POI is very small in the data. Without enough reviews, one can hardly tell which aspect is good for the POI and which is bad. In contrast, the Phoenix POIs contain around 9 times the reviews than those in Singapore, which means SAR can take advantage of the aspect and sentiment analysis to improve recommendation results.

6.4.6 Aspect Satisfaction in Regions

We explore the aspect satisfaction in some regions in the Singapore dataset. The goal of this experiment is to show the satisfaction of aspects in the regions. We compute $p(a|s, c, r)$ described in Section 6.3.3 to select most satisfied and dissatisfied aspects for

Table 6.3: User Aspect Preference & Positive Aspects of Recommended POIs

User	Preference		POI	Positive Aspect	
	Aspect	Prob		Aspect	Prob
64	menu	0.29	Paradise Dynasty (Restaurant)	general	0.98
	flavor	0.11		taste	0.96
	food	0.07		food	0.89
	quality	0.07		flavor	0.74
	service	0.07		menu	0.69
121	environment	0.36	Wala Wala Cafe Bar	quality	1
	general	0.11		location	1
	facility	0.10		taste	1
	traffic	0.10		flavor	0.97
	location	0.09		environment	0.87
420	facility	0.29	Marina Bay Sands (Hotel)	quality	1
	service	0.20		facility	0.84
	general	0.09		general	0.80
	traffic	0.07		environment	0.70
	quality	0.07		food	0.56

analysis. We show the satisfaction on “service” of Beauty & Spas POIs in two regions in Table 6.4. The first region has the highest probability of having negative overall sentiment on the aspect “service”, while the second one has the highest probability of having positive sentiment on “service”. In different regions, the sentiment on different categories is different. As shown in the example, the service is negative in Region 1, but is positive in Region 2.

Table 6.4: Aspect satisfaction of “Service” of “Beauty & Spas” in two regions

Region	Sentiment	Probability	Representative Review
1	negative	0.39	I had an oily scalp...decided to try out Yun Nam Haircare... I immediately disliked how they automatically preyed on the chinks in our armor by pin pointing the problems that we had, ...
2	positive	0.32	Nail Bar @ Cluny is a small nail salon ... I like it that it is clean and the nail technicians are professional and friendly...

6.4.7 Discussion on Topical Regions

In Table 6.5, we show the top 15 representative words with the highest probability (i.e., $p(w|r)$) for three regions in Singapore discovered by our model. The first region “Or-

Figure 6.7: The region preference of user 64 and the recommended POIs

Table 6.5: Top 15 representative words for three regions

Region	Representative Words
Orchard	place, food, really, singapore, store , shop , mall , time, pretty, shopping , quite, products , japanese, little, orchard
Changi	food, airport , singapore, place, toast, really, time, changi , pretty, terminal , kaya, coffee, service, little, free
Sentosa	place, food, sentosa , singapore, beach , really, pretty, quite, time, day, island , little, chicken, ride, people

chard” which is the largest shopping area in Singapore, contains words “store”, “shop”, “mall”, “orchard”, etc. The second region is near Changi Airport, so that words like “airport”, “changi”, “terminal”, etc. are often used in that region. The third one contains words like “sentosa”, “beach”, “island”, etc. These words are about Sentosa, which is the most famous island in Singapore. The word distribution in the three regions are characterized by spatial information.

We choose two aspects with positive and negative sentiments, and show top 20 words with highest $p(w|a, s)$ for each aspect and sentiment pair in Table 6.6. Different from region words, aspect words are like “service”, “staff”, “waiting”, etc. and sentiment words are like “good”, “best”, “bad”, etc. These words describe the aspects of the POIs and the sentiment to the aspects, while region words are more likely to be location words. As

(a) Pre@N - Phoenix

(b) Rec@N - Phoenix

(c) Pre@N - Singapore

(d) Rec@N - Singapore

Figure 6.8: Precision@N and Recall@N for user recommendation

shown in Figure 6.1, aspects are constrained by categories while regions are constrained by coordinates.

6.5 Summary

In this work, we present a novel generative model that considers aspect, sentiment and region. We investigated the related topics including geographical topic modeling, sentiment analysis and POI recommendation. To the best of our knowledge, there is no existing approach considers aspect, sentiment, category and coordinates in a unified framework.

In Section 6.2, we propose a unified model to make use of the interdependencies

Figure 6.9: MAP for user recommendation

Table 6.6: Representative words for aspect and sentiment

Aspect	Sentiment	Representative Words
waiting time	positive	time, little, long, shop, best, good, great, enjoy, come, people, place, food, like, worth, day, lunch, really, perfect, try, way
waiting time	negative	time, little, come, times, went, second, shop, wrong, like, food, place, came, getting, told, table, waiting, disappointed, eat, review, bad
service	positive	great, service, good, food, staff, place, friendly, excellent, restaurant, really, nice, location, love, pretty, like, selection, experience, little, prices, hotel
service	negative	service, order, food, minutes, place, wait, staff, time, times, bad, table, really, took, long, terrible, pretty, water, waitress, came, restaurant

among aspect, sentiment, category and region. Our model combines opinion mining and geographical modeling which has not been considered in the existing works. Both of the two aspects are considered to be important, and they interact with each other to improve the quality of each part. As discussed in Section 6.3, our model can be easily applied to several application, such as POI recommendation and user recommendation. In Section 6.4, we show that our model is effective and efficient in experiments over the applications. We also explore the reason why we recommend an item from both perspectives of aspect and region.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this thesis, we develop several techniques to address the limitations in mining topics of both individuals and regions.

Chapter 3 focuses on the problem of efficient topic exploration in the user specified region and time span. Specifically, we propose a query of Topic Mining over Documents within a Specified Region and a Timespan (TMRT). We propose a framework that combines data management principles to improve the efficiency of machine learning/data mining algorithms. It comprises of an offline pre-computation component and an online combining component. In the offline component, we divide the geo-textual documents using Octree, and pre-train LDA models on the cells of the Octree. In the online component, we propose a combining algorithm with bounded errors to combine two pre-trained LDA models and apply the combining algorithm to obtain the topics for documents in the query region and time span. Experimental results show that our framework is able to improve the runtime by at least an order of magnitude compared with the baseline methods, e.g., LDA and Online-LDA.

In Chapter 4, we conduct the first study on distributed solution of training geographical topic models. We propose three design objectives: (1) parallelism between computation and network communication; (2) reduce memory requirement; and (3) reduce network communication cost. To address these objectives, we first propose a novel training algorithm and model parallelism to allow parallelism between computation and network communication while reducing the memory requirement of each worker machine. Then, we develop two methods to reduce the network communication cost by localizing the parameters. One of them is an on-demand fetching algorithm which allows each worker machine only retrieves the parameters as needed. The other method is to allocate the parameters to related worker machines to increase in-machine access while reducing the access via network. In addition, we also propose an algorithm to reduce

the training complexity of geographical topic model, to further improve the computation efficiency. Experimental results demonstrate that our model is scalable w.r.t. number of machines, data size and model size, and outperforms baseline solutions by several orders of magnitude.

In Chapter 5, we study the problem of annotating points of interest (POIs) with geo-tagged microblogs to provide a complementary resource for mining topics of individuals. We propose a supervised Bayesian model, which exploits the textual content, coordinates, and user behaviors. We make use of these three factors under the supervision information, i.e., a POI-relatedness variable to distinguish the POI-related microblogs to those are not related to POI. We also extend the model by integrating external text sources of POIs, e.g., Foursquare tips, to enhance the annotation accuracy. The results of experimental studies show that the proposed supervised Bayesian model is able to associating POIs with geo-tagged tweets effectively. It outperforms the second best baselines by 28.3% and 20.4% in terms of F1 score in the New York city dataset and Singapore dataset.

In Chapter 6, we study the problem of mining user interests on topical-regions, topical-aspects and categories. We propose a novel unified model, namely Sentiment-Aspect-Region (SAR). The SAR model is able to make use of the interdependencies among aspect, sentiment, category and region. It combines the techniques of opinion mining and geographical topic modeling. We also develop an efficient algorithm to provide efficient recommendations based on the SAR model, and present several applications of the proposed model including POI recommendation, user recommendation, explaining the recommendation results, and aspect satisfaction analysis in regions. Experimental results show the SAR model improves the mean average precision (MAP) by 70% and 44% in POI recommendations and significantly outperforms the baselines in user recommendation.

My research on the aforementioned problems has significant impact in both research and industry. Firstly, the proposed topic exploration solution is the first to combine the ideas of spatial data management and machine learning in topic modeling, and thus shows the possibility of solving other data mining problems efficiently by leveraging the spatial data management methods. It is also beneficial for researchers in other disciplines such as marketing and social science to analyze the market share and events in regions, respectively. Secondly, based on the findings in the distributed training problem of geo-topic models, data scientist can extend our distributed training algorithm to existing graphical models for user behavior modeling, POI recommendations, and event detection, in order to fit data with industrial scale. Thirdly, by applying the proposed POI annotation method, the POI dataset can be enriched by hundreds of millions of geo-tagged tweets, providing a great opportunity to improve the accuracy of recommendation and event detection models. Finally, the proposed probabilistic model for mining user interests on aspects and regions creates high commercial value because it does not

only provide a new way for business owner to analyze the satisfaction of users but also improve the user experience by providing explanation of recommendation results. As a result, a sound business cycle between merchants and users can be established.

7.2 Future Directions

My research opens several potential directions for future work.

- Our work on topic mining and exploration in regions combines data management principle to machine learning algorithms, which inspires a number of promising directions for future work. Firstly, to support exploring up-to-date topic from streaming data, it is important to efficiently learn topics from the stream and update the pre-trained hierarchy. Secondly, topics are not static overtime. It is interesting to explore the trends and evolution of topics in a region. Thirdly, users may want to explore topics that are related to some specified keywords. Therefore, extending the TMRT query with keywords as an additional query dimension is another future direction.
- Apart from geographical topic models, many effective probabilistic models on mining geo-textual data have been designed. These models consider more factors than topics (e.g., aspects, sentiment, time, etc.), but are often intractable and inefficient for training. Given the availability of high performance computation clusters, it is an opportunity to develop distributed solutions for training models with other latent variables, e.g., time and sentiment. In addition, there exists other formulations of geographical topic models (e.g., sparse additive model) that are not considered in our work. Another future direction can be extending the proposed distributed solution PGeoTopic to support different formulations of geographical topic models.
- Given that social media posts such as tweets are often generated in the streaming manner at a high rate, it is interesting to build a model that not only supports on-line updating but also can efficiently associate streaming documents to their related POIs. In addition, our work mainly focus on geo-tagged microblogs, while most of the existing methods work on documents without geo-tags [KMO11, DKP12, LHFT14, LPC⁺15]. It is an open problem how to annotate these two types of documents in a unified way such that the modeling of the two types of documents can enhance each other. For example, after some geo-tagged documents are associated to POIs, they can be used for computing semantic relatedness between POIs and the documents without geo-tags.

- For the problem of mining topics of individuals (e.g., user interests on topics), we develop a unified model for mining topical aspects and topical regions and the user's interests on aspects, regions and categories. Because users may vary their interests on aspects, regions and categories over time, temporal information should also be important. However, there exist no unified solution that considers all the factors (i.e., time, aspect, region, category, and sentiment). It would be interesting to develop time-aware models for capturing the dynamic of user interests over time.

Appendix A

List of Publications

- Daniel Rugeles, **Kaiqi Zhao**, Gao Cong, Manoranjan Dash, Shonali Krishnaswamy. Biclustering: An application of Dual Topic Models. Proceedings of the 2017 SIAM International Conference on Data Mining (**SDM 2017**), pages 453-461, Houston, TX, USA, April 27-29, 2017.
- **Kaiqi Zhao**, Gao Cong, Aixin Sun. Annotating Points of Interest with Geo-tagged Tweets. Proceedings of the 25th ACM International Conference on Information and Knowledge Management (**CIKM 2016**), pages 417-426, Indianapolis, IN, USA, October 24-28, 2016.
- **Kaiqi Zhao**, Lisi Chen, Gao Cong. Topic Exploration in Spatio-Temporal Document Collections. Proceedings of the 2016 International Conference on Management of Data (**SIGMOD 2016**), pages 985-998, San Francisco, CA, USA, June 26 - July 01, 2016.
- **Kaiqi Zhao**, Yiding Liu, Quan Yuan, Lisi Chen, Zhida Chen, Gao Cong. Towards Personalized Maps: Mining User Preferences from Geo-textual Data (Demo). Proceedings of the VLDB Endowment (**PVLDB 2016**), Volume 9, Issue 13, pages 1545-1548, 2016.
- Kaiyu Feng, **Kaiqi Zhao**, Yiding Liu, Gao Cong. A System for Region Search and Exploration (Demo). Proceedings of the VLDB Endowment (**PVLDB 2016**), Volume 9, Issue 13, pages 1549-1552, 2016.
- Quan Yuan, Gao Cong, **Kaiqi Zhao**, Zongyang Ma, Aixin Sun. Who, Where, When and What: a Non-parametric Bayesian Approach to Context-aware Recommendation and Search for Twitter Users. ACM Transactions on Information Systems (**TOIS**), Volume 33 Issue 1, Article No. 2, March 2015.

- **Kaiqi Zhao**, Gao Cong, Quan Yuan, Kenny Q. Zhu. SAR: A sentiment-aspect-region model for user preference analysis in geo-tagged reviews. 31st IEEE International Conference on Data Engineering (**ICDE 2015**), Seoul, South Korea, April 13-17, 2015.

References

- [AAG⁺12] Amr Ahmed, Moahmed Aly, Joseph Gonzalez, Shravan Narayanamurthy, and Alexander J. Smola. Scalable inference in latent variable models. In *WSDM*, pages 123–132, 2012.
- [ABD08] Loulwah AlSumait, Daniel Barbará, and Carlotta Domeniconi. On-line LDA: adaptive topic models for mining text streams with applications to topic detection and tracking. In *ICDM*, pages 3–12, 2008.
- [AHS13] Amr Ahmed, Liangjie Hong, and Alexander J. Smola. Hierarchical geographical modeling of user locations from social media posts. In *WWW*, pages 25–36, 2013.
- [AKSS09] Albert Angel, Nick Koudas, Nikos Sarkas, and Divesh Srivastava. What’s on the grapevine? In *SIGMOD*, pages 1047–1050, 2009.
- [AMH⁺15] Ahmed M Aly, Ahmed R Mahmood, Mohamed S Hassan, Walid G Aref, Mourad Ouzzani, Hazem Elmeleegy, and Thamir Qadah. Aqwa: adaptive query workload aware partitioning of big spatial data. *Proceedings of the VLDB Endowment*, 8(13):2062–2073, 2015.
- [BKB15] Tanusri Bhattacharya, Lars Kulik, and James Bailey. Automatically recognizing places of interest from unreliable gps data using spatio-temporal density estimation and line intersections. *Pervasive Mob. Comput.*, 19(C):86–107, 2015.
- [BM07] David M. Blei and Jon D. McAuliffe. Supervised topic models. In *NIPS*, pages 121–128, 2007.
- [BNJ03] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

REFERENCES

- [CLZC16] Jianfei Chen, Kaiwei Li, Jun Zhu, and Wenguang Chen. Warplda: a cache efficient o (1) algorithm for latent dirichlet allocation. *Proceedings of the VLDB Endowment*, 9(10):744–755, 2016.
- [CML11] Eunjoon Cho, Seth A. Myers, and Jure Leskovec. Friendship and mobility: User movement in location-based social networks. In *SIGKDD*, pages 1082–1090, 2011.
- [DKP12] Nilesh Dalvi, Ravi Kumar, and Bo Pang. Object matching in tweets with spatial models. In *WSDM*, pages 43–52, 2012.
- [DZL⁺11] Bolin Ding, Bo Zhao, Cindy Xide Lin, Jiawei Han, Chengxiang Zhai, Ashok N. Srivastava, and Nikunj C. Oza. Efficient keyword-based search for top-k cells in text cube. *IEEE Trans. Knowl. Data Eng.*, 23(12):1795–1810, 2011.
- [EM15] Ahmed Eldawy and Mohamed F Mokbel. Spatialhadoop: A mapreduce framework for spatial data. In *ICDE*, pages 1352–1363, 2015.
- [EOSX10] Jacob Eisenstein, Brendan O’Connor, Noah A. Smith, and Eric P. Xing. A latent variable model for geographic lexical variation. In *EMNLP*, pages 1277–1287, 2010.
- [FCH⁺08] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874, June 2008.
- [FLN01] Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. In *PODS*, pages 102–113, 2001.
- [FZZ⁺15] Wei Feng, Chao Zhang, Wei Zhang, Jiawei Han, Jianyong Wang, Charu Agarwal, and Jianbin Huang. STREAMCUBE: hierarchical spatio-temporal hashtag clustering for event exploration over the twitter stream. In *ICDE*, pages 1561–1572, 2015.
- [GG16] Jinjin Guo and Zhiguo Gong. A nonparametric model for event discovery in the geospatial-temporal space. In *CIKM*, pages 499–508, 2016.

REFERENCES

- [HAG⁺12] Liangjie Hong, Amr Ahmed, Siva Gurumurthy, Alexander J. Smola, and Kostas Tsioutsoulis. Discovering geographical topics in the twitter stream. In *WWW*, pages 769–778, 2012.
- [Has70] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.
- [HCC11] Lichan Hong, Gregorio Convertino, and Ed H. Chi. Language matters in twitter: A large scale study. In *Proceedings of the Fifth International Conference on Weblogs and Social Media, Barcelona, Catalonia, Spain, July 17-21, 2011*, 2011.
- [HCC⁺13] Qirong Ho, James Cipar, Henggang Cui, Seunghak Lee, Jin Kyu Kim, Phillip B Gibbons, Garth A Gibson, Greg Ganger, and Eric P Xing. More effective distributed ml via a stale synchronous parallel parameter server. In *NIPS*, pages 1223–1231, 2013.
- [HE13] Bo Hu and Martin Ester. Spatial topic modeling in online social media for location recommendation. In *RecSys*, pages 25–32, 2013.
- [HJE13] Bo Hu, Mohsen Jamali, and Martin Ester. Spatio-temporal topic modeling in mobile social media for location recommendation. In *ICDM*, pages 1073–1078, 2013.
- [HK00] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.
- [JO11] Yohan Jo and Alice H. Oh. Aspect and sentiment unification model for online review analysis. In *WSDM*, pages 815–824, 2011.
- [Joa06] Thorsten Joachims. Training linear svms in linear time. In *SIGKDD*, pages 217–226, 2006.
- [KIH⁺13] Takeshi Kurashima, Tomoharu Iwata, Takahide Hoshide, Noriko Takaya, and Ko Fujimura. Geo topic model: Joint modeling of user’s activity area and interests for location recommendation. In *WSDM*, pages 375–384, 2013.
- [KMO11] Sheila Kinsella, Vanessa Murdock, and Neil O’Hare. ”i’m eating a sandwich in glasgow”: Modeling locations with tweets. In *SMUC*, pages 61–68, 2011.

REFERENCES

- [LAP⁺14] Mu Li, David G Andersen, Jun Woo Park, Alexander J Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J Shekita, and Bor-Yiing Su. Scaling distributed machine learning with the parameter server. In *OSDI*, volume 14, pages 583–598, 2014.
- [LARS14] Aaron Q. Li, Amr Ahmed, Sujith Ravi, and Alexander J. Smola. Reducing the sampling complexity of topic models. In *KDD*, pages 891–900, 2014.
- [LCDZ11] Yue Lu, Malu Castellanos, Umeshwar Dayal, and ChengXiang Zhai. Automatic construction of a context-aware sentiment lexicon: An optimization approach. In *WWW*, pages 347–356, 2011.
- [LDH⁺08] Cindy Xide Lin, Bolin Ding, Jiawei Han, Feida Zhu, and Bo Zhao. Text cube: Computing IR measures for multidimensional text database analysis. In *ICDM*, pages 905–910, 2008.
- [LEHC15] Yu Liu, Martin Ester, Bo Hu, and David W. Cheung. Spatio-temporal topic models for check-in data. In *ICDM*, pages 889–894, 2015.
- [LFYX13] Bin Liu, Yanjie Fu, Zijun Yao, and Hui Xiong. Learning geographical preferences for point-of-interest recommendation. In *KDD*, pages 1043–1051, 2013.
- [LHFT14] Guoliang Li, Jun Hu, Jianhua Feng, and Kian-Lee Tan. Effective location identification from microblogs. In *ICDE*, pages 880–891, 2014.
- [Liu94] Jun S Liu. The collapsed gibbs sampler in bayesian computations with applications to a gene regulation problem. *Journal of the American Statistical Association*, 89(427):958–966, 1994.
- [LN89] D. C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Math. Program.*, 45(3):503–528, December 1989.
- [LPC⁺15] Xutao Li, Tuan-Anh Nguyen Pham, Gao Cong, Quan Yuan, Xiao-Li Li, and Shonali Krishnaswamy. Where you instagram?: Associating your instagram photos with points of interest. In *CIKM*, pages 1231–1240, 2015.
- [LX11] Defu Lian and Xing Xie. Learning location naming from user check-in histories. In *GIS*, pages 112–121, 2011.

REFERENCES

- [LX14] Defu Lian and Xing Xie. Mining check-in history for personalized location naming. *ACM Trans. Intell. Syst. Technol.*, 5(2):32:1–32:25, April 2014.
- [LXP⁺15] Bin Liu, Hui Xiong, Spiros Papadimitriou, Yanjie Fu, and Zijun Yao. A general geographical probabilistic factor model for point of interest recommendation. *TKDE*, 27(5):1167–1179, 2015.
- [LZCS11] Zhiyuan Liu, Yuzhou Zhang, Edward Y. Chang, and Maosong Sun. Plda+: Parallel latent dirichlet allocation with data placement and pipeline processing. *ACM Trans. Intell. Syst. Technol.*, 2(3):26:1–26:18, May 2011.
- [ME11] Samaneh Moghaddam and Martin Ester. Ilda: Interdependent lda model for learning latent aspects and their ratings from online product reviews. In *SIGIR*, pages 665–674, 2011.
- [Mea82] Donald Meagher. Geometric modeling using octree encoding. *Computer Graphics and Image Processing*, 19(2):129–147, 1982.
- [MLW⁺07] Qiaozhu Mei, Xu Ling, Matthew Wondra, Hang Su, and ChengXiang Zhai. Topic sentiment mixture: Modeling facets and opinions in weblogs. In *WWW*, pages 171–180, 2007.
- [MTW⁺04] George Marsaglia, Wai Wan Tsang, Jingbo Wang, et al. Fast generation of discrete random variables. *Journal of Statistical Software*, 11(3):1–11, 2004.
- [NASW09] David Newman, Arthur Asuncion, Padhraic Smyth, and Max Welling. Distributed algorithms for topic models. *Journal of Machine Learning Research*, 10(Aug):1801–1828, 2009.
- [NTKS14] Kyosuke Nishida, Hiroyuki Toda, Takeshi Kurashima, and Yoshihiko Suhara. Probabilistic identification of visited point-of-interest for personalized automatic check-in. In *UbiComp*, pages 631–642, 2014.
- [SAKS10] Nikos Sarkas, Albert Angel, Nick Koudas, and Divesh Srivastava. Efficient identification of coupled entities in document collections. In *ICDE*, pages 769–772, 2010.
- [SBSR08] Alkis Simitsis, Akanksha Baid, Yannis Sismanis, and Berthold Reinwald. Multidimensional content exploration. *PVLDB*, 1(1):660–671, 2008.

REFERENCES

- [Siz10] Sergej Sizov. Geofolk: Latent spatial semantics in web 2.0 social media. In *WSDM*, pages 281–290, 2010.
- [SM15] Luke Sloan and Jeffrey Morgan. Who tweets with their location? understanding the relationship between demographic characteristics and the use of geoservices and geotagging on twitter. *PloS one*, 10(11):e0142209, 2015.
- [SN10] Alexander Smola and Shravan Narayanamurthy. An architecture for parallel topic models. *PVLDB*, 3(1-2):703–710, September 2010.
- [SSSH13] Blake Shaw, Jon Shea, Siddhartha Sinha, and Andrew Hogue. Learning to rank for spatiotemporal search. In *WSDM*, pages 717–726, 2013.
- [SW85] Hanan Samet and Robert E. Webber. Storing a collection of polygons using quadrees. *ACM Trans. Graph.*, 4(3):182–222, 1985.
- [Tie94] Luke Tierney. Markov chains for exploring posterior distributions. *the Annals of Statistics*, pages 1701–1728, 1994.
- [TM08a] Ivan Titov and Ryan McDonald. Modeling online reviews with multi-grain topic models. In *WWW*, pages 111–120, 2008.
- [TM08b] Ivan Titov and Ryan T. McDonald. A joint model of text and aspect ratings for sentiment summarization. In *ACL*, pages 308–316, 2008.
- [WB11] Benjamin P. Wing and Jason Baldridge. Simple supervised document geolocation with geodesic grids. In *HLT*, pages 955–964, 2011.
- [WBS⁺09] Yi Wang, Hongjie Bai, Matt Stanton, Wen-Yen Chen, and Edward Y. Chang. Plda: Parallel latent dirichlet allocation for large-scale applications. In *AAIM*, pages 301–314, 2009.
- [WLZ10] Hongning Wang, Yue Lu, and Chengxiang Zhai. Latent aspect rating analysis on review text data: A rating regression approach. In *KDD*, pages 783–792, 2010.
- [WMSM09] Hanna M. Wallach, Iain Murray, Ruslan Salakhutdinov, and David Mimno. Evaluation methods for topic models. In *ICML*, pages 1105–1112, 2009.

REFERENCES

- [WYC⁺15] Weiqing Wang, Hongzhi Yin, Ling Chen, Yizhou Sun, Shazia Sadiq, and Xiaofang Zhou. Geo-sage: A geographical sparse additive generative model for spatial item recommendation. In *KDD*, pages 1255–1264, 2015.
- [WZS⁺14] Yi Wang, Xuemin Zhao, Zhenlong Sun, Hao Yan, Lifeng Wang, Zhihui Jin, Liubin Wang, Yang Gao, Jia Zeng, Qiang Yang, et al. Towards topic modeling for big data. *arXiv preprint arXiv:1405.4402*, 2014.
- [XLY⁺16] Dong Xie, Feifei Li, Bin Yao, Gefei Li, Liang Zhou, and Minyi Guo. Simba: Efficient in-memory spatial analytics. In *SIGMOD*, pages 1071–1085, 2016.
- [YCC⁺14a] Hongzhi Yin, Bin Cui, Ling Chen, Zhiting Hu, and Zi Huang. A temporal context-aware model for user behavior modeling in social media systems. In *SIGMOD*, pages 1543–1554, 2014.
- [YCC⁺14b] Hongzhi Yin, Bin Cui, Ling Chen, Zhiting Hu, and Chengqi Zhang. Modeling location-based user rating profiles for personalized recommendation. *TKDD*, January 2014.
- [YCH⁺11] Zhijun Yin, Liangliang Cao, Jiawei Han, Chengxiang Zhai, and Thomas Huang. Geographical topic discovery and comparison. In *WWW*, WWW ’11, pages 247–256, 2011.
- [YCM⁺13] Quan Yuan, Gao Cong, Zongyang Ma, Aixin Sun, and Nadia Magnenat Thalmann. Who, where, when and what: Discover spatio-temporal topics for twitter users. In *SIGKDD*, pages 605–613, 2013.
- [YCZ⁺15] Quan Yuan, Gao Cong, Kaiqi Zhao, Zongyang Ma, and Aixin Sun. Who, where, when, and what: A nonparametric bayesian approach to context-aware recommendation and search for twitter users. *ACM Trans. Inf. Syst.*, 33(1):2:1–2:33, 2015.
- [YGH⁺15] Jinhui Yuan, Fei Gao, Qirong Ho, Wei Dai, Jinliang Wei, Xun Zheng, Eric Po Xing, Tie-Yan Liu, and Wei-Ying Ma. Lightlda: Big topic models on modest computer clusters. In *WWW*, pages 1351–1361. ACM, 2015.
- [YHY⁺15] Hsiang-Fu Yu, Cho-Jui Hsieh, Hyokun Yun, SVN Vishwanathan, and Inderjit S Dhillon. A scalable asynchronous distributed algorithm for topic

REFERENCES

- modeling. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1340–1350. International World Wide Web Conferences Steering Committee, 2015.
- [YMM09] Limin Yao, David Mimno, and Andrew McCallum. Efficient methods for topic model inference on streaming document collections. In *KDD*, pages 937–946, 2009.
- [YSC⁺13] Hongzhi Yin, Yizhou Sun, Bin Cui, Zhiting Hu, and Ling Chen. LCARS: a location-content-aware recommender system. In *KDD*, pages 221–229, 2013.
- [YWS16] Jia Yu, Jinxuan Wu, and Mohamed Sarwat. A demonstration of geospark: A cluster computing framework for processing big spatial data. In *ICDE*, pages 1410–1413, 2016.
- [YYLL11] Mao Ye, Peifeng Yin, Wang-Chien Lee, and Dik-Lun Lee. Exploiting geographical influence for collaborative point-of-interest recommendation. In *SIGIR*, pages 325–334, 2011.
- [YZG15] Simin You, Jianting Zhang, and Le Gruenwald. Large-scale spatial join query processing in cloud. In *ICDE Workshops*, pages 34–41, 2015.
- [YZS⁺15] Hongzhi Yin, Xiaofang Zhou, Yingxia Shao, Hao Wang, and Shazia Sadiq. Joint modeling of user check-in behaviors for point-of-interest recommendation. In *CIKM*, pages 1631–1640, 2015.
- [ZCYZ15] Kaiqi Zhao, Gao Cong, Quan Yuan, and Kenny Q. Zhu. Sar: A sentiment-aspect-region model for user preference analysis in geo-tagged reviews. In *ICDE*, pages 675–686, 2015.
- [ZHSS16] Cong Zheng, E Haihong, Meina Song, and Junde Song. Tgtm: Temporal-geographical topic model for point-of-interest recommendation. In *International Conference on Database Systems for Advanced Applications*, pages 348–363. Springer, 2016.
- [ZLDH11] Bo Zhao, Cindy Xide Lin, Bolin Ding, and Jiawei Han. Texplorer: keyword-based object search and exploration in multidimensional text databases. In *CIKM*, pages 1709–1718, 2011.

REFERENCES

- [ZLZ⁺14] Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *SIGIR*, pages 83–92, 2014.
- [ZR13] Ce Zhang and Christopher Ré. Towards high-throughput gibbs sampling at scale: A study across storage managers. In *SIGMOD*, pages 397–408, 2013.
- [ZZH09] Duo Zhang, ChengXiang Zhai, and Jiawei Han. Topic cube: Topic modeling for OLAP on multidimensional text databases. In *SDM*, pages 1124–1135, 2009.
- [ZZH11] Duo Zhang, ChengXiang Zhai, and Jiawei Han. Mitexcube: Microtextcluster cube for online analysis of text cells. In *CIDU*, pages 204–218, 2011.
- [ZZZ⁺14] Yongfeng Zhang, Haochen Zhang, Min Zhang, Yiqun Liu, and Shaoping Ma. Do users rate or review?: Boost phrase-level sentiment labeling with review-level sentiment classification. In *SIGIR*, pages 1027–1030, 2014.