This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.2989142, IEEE Transactions on Knowledge and Data Engineering

PGeoTopic: A Distributed Solution for Mining Geographical Topic Models

Kaiqi Zhao, Gao Cong, and Xiucheng Li

Abstract—Geographical topic models have been used to mine geo-tagged documents for topical region and geographical topics, and also have applications in recommendations, user mobility modeling, event detection, etc. Existing studies focus on learning effective geographical topic models while ignoring the efficiency issue. However, it is very expensive to train geographical topic models — it may take days to train a geographical topic model of a small scale on a collection of documents with millions of word tokens. In this paper, we propose the first distributed solution, called PGeoTopic, for training geographical topic models. The proposed solution comprises several novel technical components to increase parallelism, reduce memory requirement, and reduce communication cost. Experiments show that our approach for mining geographical topic models is scalable with both model size and data size on distributed systems.

Index Terms—Geographical topic model, Distributed machine learning

1 Introduction

With the prominence of GPS-equipped device, a huge amount of documents associated with geo-locations (e.g., coordinates) are increasingly generated on the Web, such as geo-tagged tweets, webpages and reviews. These geo-textual documents contain both massive useful information on what topics were discussed and where they were discussed. It is of great interest to mine such data to answer questions like how is topic "education" distributed over United States and what are the differences between Redmond and Silicon Valley in topics? In addition, mining geo-textual data opens many research areas, such as point-of-interest recommendation [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], event detection [11], [12], geotagging [13], [14] and travel planning [15], [16], etc.

To mine geo-textual documents, geographical topic models are proposed [1], [2], [3], [4], [6], [7], [10], [12], [17], [18], [19], [20], [21]. There are two main concepts of such models, i.e., "geographical topics" and "topical regions" [18]. Intuitively, geographical topics are spatially coherent topics, e.g., "education" in a university region. Topical region is a group of nearby locations that share similar topic distributions. Given a collection of geo-textual documents, geographical topic models learn the geographical topics and topical regions in an unsupervised manner. Each topic is often modeled as a word distribution. Each region contains two components – (1) a topic distribution to model its semantics; and (2) a Gaussian distribution over locations to model its spatial information [2], [3], [10], [12], [18], [19]. Example 1.1 shows an example geographical topic model.

Example 1.1. *Let $D = \{d_1, d_2, ..., d_{10}\}$ be a set of geo-tagged documents. Figure 1 illustrates their locations. The word tokens of each document are listed in Table 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 1 and returns the topical regions r_1 and r_2 in Figure 1. Each region has a topic distribution, e.g., region r_1 is mainly about the movie topic, and its spatial information is represented by a Gaussian distribution. Each topic is described by a distribution of words. In the example, the movie topic is represented by the distribution $\langle$"movie" $= \frac{1}{8}$, "cold" $= \frac{1}{8}$, "3D" $= \frac{1}{16}, ...\rangle$. 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". The ovals in Figure 1 represent the contour lines of the Gaussian distributions.*

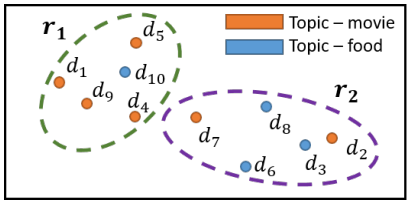


Fig. 1: An example of geographical topics

TABLE 1: Word tokens of documents. The topic estimated for each word token is placed in brackets. 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)

Although the model quality of geographical topic models is extensively studied in the existing literature [2], [3], [10], [12], [18], [19], the training efficiency is an important

issue for learning big models from big data. 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, resulting in a model with more than 40 billion parameters (up to 160 GB[1]). Training such a big model on hundreds of millions of documents may take months on a modern machine. Worse still, we cannot simply divide the map into grids and train small topic models independently in each grid due to three reasons – (1) It is non-trivial to estimate the geographical distribution of a given topic across the subareas as the latent topics learned in different subareas are often not comparable. (2) Improper division of the data may split topical regions on the boundary of subareas and these subareas cannot be easily merged afterward. (3) Learning a model in each subarea results in local optimal models that are not globally optimized to the whole dataset.

To meet the need of learning geographical topic models on big data, it is of great interest to utilize powerful computer clusters. Recently, distributed solutions [22], [23], [24], [25], [26], [27], [28], [29] for topic models are proposed. However, it is difficult to apply the existing solutions to geographical topic models. Different from topic models, geographical topic models have two big matrices of parameters that may contain tens of billions of entries, i.e., the **word-topic distribution matrix** and the **topic-region distribution matrix**, and these two parameter matrices are highly coupled in collapsed Gibbs sampling, the most popular method to train geographical topic models [1], [3], [4], [10], [12], [17], [19]. The existing solutions for topic models only consider the word-topic distribution matrix, lacking support of efficient synchronization for both matrices. Therefore, they are not applicable for geographical topic models.

In this work, we propose PGeoTopic, the first solution to Parallelize the training of Geographical Topic models over multiple machines. In general, we consider the following design 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 needs both matrices of parameters due to their interdependency, and thus all the parameters have to be fetched from servers before sampling. Such a requirement makes it difficult to parallelize the sampling computation and the communication, and also incurs expensive requirement on memory. Second, the two matrices need to be distributed to the worker machines if the parameter server framework [30], [31], [32], the most popular distributed machine learning architecture, is adopted. 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, it is difficult to achieve parameter localization and fetching parameters on-demand.

PGeoTopic follows the setting of parameter servers [30], [31], [32] in which the two matrix parameters are distributed and stored in each worker's shared memory. To address the above challenges, PGeoTopic comprises the following novel techniques. Firstly, to support parallelism while reducing memory requirement, we propose a new training algorithm that decouples the interdependency of the word-topic matrix and the topic-region matrix to allow independent access to them. We also design a model parallelism for the proposed training algorithm that partitions the two parameter matrices into slices. The model parallelism supports the parallelism of computation and network communication and allows each worker machine only work on one slice of either matrix each time. Secondly, we propose to adopt spatial partitioning to make data allocated to each worker spatially close, and propose solutions to achieve parameter localization and on-demand parameter fetching, thus reducing communication cost. In summary, this paper makes the following contributions:

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

2 Related Work

In this section, we introduce the existing distributed training methods for topic models.

2.1 Geographical Topic Models

Geographical topic modeling has been a hot research topic in recent years. Sizov et al. [17], [33] proposes the *GeoFolk* model, which introduces latent regions in geographical topic modeling. In GeoFolk, each latent region consists of two one-dimensional Gaussian distributions on latitude and longitude, respectively, and a multinomial word distribution. Yin et al. [18] introduce latent topics and assume each region has a topic distribution instead of a word distribution. This model advances GeoFolk in that it captures multiple topics for each region. Ahmed et al. [19] propose a non-parametric approach to automatically learn a hierarchy of latent regions and their corresponding topic distributions.

Recent studies focus on applying geographical topic models to social media applications, including point-of-interest (POI) recommendations [1], [6], [7], [8], [10], event detection [11], [12], geotagging [13], [14] and travel planning [15], [16], etc. Yin et al. [7] jointly model spatial, time and user information to improve POI recommendation accuracy. Guo et al. [12] improve local event detection by a non-parametric method that is able to learn the topics and regions automatically. Liu et al. [15], [16] propose to recommending travel packages by considering geographical topics in different regions. Hong et al. [13] propose a model

1. Suppose there are 300K unique words as in our Twitter dataset.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.2989142, IEEE Transactions on Knowledge and Data Engineering

to predict the coordinates of tweets without geo-tags by learning user's preferences on latent regions and topics.

Although the above existing studies show the effectiveness of geographical topics models, the training efficiency is known to be an open problem for those models. Most of the existing geographical topic models are trained using Gibbs sampling, which is a Markov Chain Monte Carlo (MCMC) method. Some studies are proposed to improve the efficiency of MCMC on single machines [34], such as Hamiltonian Monte Carlo (HMC) [35] and efficient importance sampling (EIS) [36]. However, these more efficient MCMC methods are based on strong assumptions on the target distribution and cannot be directly applied to geographical topic models. For example, HMC does not work on discrete variables while geo-topic models have both discrete and continuous variables. EIS [36] assumes the density of the target distribution to as an integral while the posterior distribution if region and topic is not an integral. No study shows the applicability of these more efficient MCMC methods to geo-topic models to the best of our knowledge.

2.2 Distributed Training of Topic Models

Our work is related to distributed training solutions to topic models [22], [24], [25], [26], [27], [28], [29], especially the Latent Dirichlet Allocation model [37]. Hierarchical Distributed LDA (HD-LDA) [28] assigns data across different processors, and each processor performs Gibbs sampling on its data followed by a global synchronization of parameters in each iteration. Ahmed et al. [22] and Li et al. [23] apply data parallel strategy, in which they partition and distribute the documents 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 learns topics from local data. However, when the model cannot 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, which is similar to the first baseline used in our experiments.

F+LDA [27] partitions the data across multiple workers and builds a binary tree structure to speed up the sampling. Peacock [24] and LightLDA [25] adopt both data parallel and model parallel strategies. Peacock partitions the model by words and each worker takes care of the words appearing in its model partition. However, the extra document-topic distributions has to be synchronized because a document is processed across workers. LightLDA partitions the model into slices by words as in Peacock, but distributes each document to a single worker. Model slices are scheduled to workers slice by slice in each iteration such that the workers process different parts of the model independently at the same time. WarpLDA [26] further improves 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. Worse still, the region-topic matrix may not be able to fit into memory. In our experiment, we extend LightLDA for training geographical topic models as a baseline. To the best of our knowledge, no existing work considers training a geographical topic model in a distributed setting.

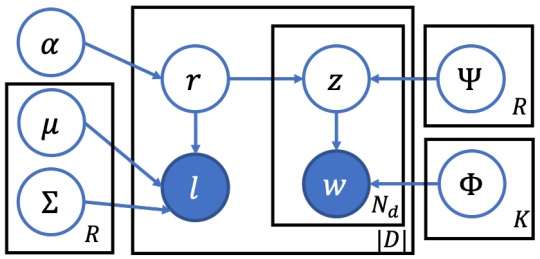


Fig. 2: Graphical representation of a geographical topic model.

3 Preliminaries

In this section, we introduce geographical topic models and the corresponding training algorithm. For easy reference, we summarize the notations in Table 2.

TABLE 2: Summary of notations

Notation	Description
D	The set of geo-tagged documents
K, R	Number of topics and number of regions
z_{di}	Topic assignment for i-th word in document d
r_d	Region assignment for document d
$\mathbf{l}_d$	Location assignment for document d
$\eta \in \mathbb{R}^R$	Region distribution
$\Psi \in \mathbb{R}^{R \times K}$	The topic-region matrix
$\Phi \in \mathbb{R}^{K \times V}$	The word-topic matrix
$\Psi_r \in \mathbb{R}^K$	The topic distribution of region r
$\Phi_z \in \mathbb{R}^V$	The word distribution of topic z
μ_r, Σ_r	The mean and variance of region r
m_{zr}	Number of co-occurrences of the topic z and region r in the same document
n_{zw}	Frequency of assigning topic z to word w
n_z	Frequency of topic z in the dataset
n_r	Frequency of region r in the dataset
α, β, γ	Hyper-parameters of the model

3.1 Geographical Topic Model

In a geographical topic model, each latent topic is often defined as a distribution of words Φ_z. Each latent region r is often defined as a Gaussian distribution $\mathcal{N}(\mu_r, \Sigma_r)$ with mean μ_r and covariance Σ_r, and a distribution of topics Ψ_r. A geospatial document is generated as follows [18]:

1) Draw region distribution $\eta \sim \text{Dir}(\alpha)$
2) Draw a latent region index $r \sim \text{Multi}(\eta)$
3) Draw a location $\mathbf{l} \sim \mathcal{N}(\mu_r, \Sigma_r)$
4) For each word w in document d,
   a) Draw a topic index $z \sim \text{Multi}(\Psi_r)$
   b) Draw $w \sim \text{Multi}(\Phi_z)$

Figure 2 shows the graphical representation of a geographical topic model. Note that some models [1], [4], [10], [12], [20], [21] may consider more variables (e.g., time) besides topics and regions. We will show that our solution can also handle models with additional variables in Section 5.6.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.2989142, IEEE Transactions on Knowledge and Data Engineering

3.2 Training Geographical Topic Models

Most of the geographical topic models are learned using Collapsed Gibbs sampling [1], [3], [4], [10], [12], [17], [19]. Specifically, a region r and a topic k is sampled for the i-th word token w in the document d with the following probability, given the topic and region assignments of other word tokens and the document collection D:

$$p(z_{di}=k, r_{di}=r|\Psi^{-di}, \Phi^{-di}, D) \propto \underbrace{\frac{n_{kw}^{-di}+\beta_{kw}}{\sum_{w'} n_{kw'}^{-di}+\beta_{kw'}}}_{\text{word-topic matrix}} \cdot \underbrace{\frac{m_{kr}^{-di}+\gamma_{kr}}{\sum_{k'} m_{k'r}^{-di}+\gamma_{k'r}}}_{\text{topic-region matrix}} \underbrace{f(\mathbf{l}_d|\mu_r,\Sigma_r)\cdot(n_r^{-d}+\alpha_r)}_{\text{location-region relation}} \quad (1)$$

where n_{kw}^{-di} denotes the frequency of assigning word w to topic k excluding the assignment of the i-th word token in document d. The notation m_{kr}^{-di} denotes the frequency of assigning topic k to region r excluding the assignment of the current token. The hyperparamters α_r, β_{kw}, 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, respectively[2]. The function $f(\cdot|\mu_r,\Sigma_r)$ denotes the probability density of the Gaussian distribution of region r. The count n_r^{-d} is the number of documents falling into region r excluding document d. After the regions of word tokens in a document are sampled, the region assignment of a word token is randomly selected as the region of the document. The training process runs iteratively until the model converges.

The first two terms in Eq. (1) can be stored as matrices. For ease of presentation, we use **word-topic matrix** denoted by Φ and **topic-region matrix** denoted by Ψ to refer to the two parameters in the remaining of the paper. The Gibbs sampler needs to access the whole topic-region matrix when computing Eq. (1) for all topic-region pairs. The whole topic-region matrix can be too large to be synchronized among machines in a distributed setting and may not even fit into the local memory.

4 Problem Statement

This paper focuses on geo-textual documents, such as geo-tagged tweets, Foursquare check-ins, geo-tagged news, geo-tagged web pages, etc. Formally, we define geo-textual document as follows.

Definition 1. ***Geo-textual Document****. A geo-textual document is a tuple $d = \langle \boldsymbol{w}_d, \boldsymbol{l}_d \rangle$, where $\boldsymbol{w}_d$ is a set of word tokens and $\boldsymbol{l}_d$ is a latitude-longitude geo-location.*

Consider that the data is often much larger than the model, we adopt the parameter server setting [30], [31], [32] as does the recent study on distributed LDA [25], [26]. That is, we transfer the model instead of the data via network during training. In the parameter server architecture, the memory of each machine is divided into 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, we formulate the problem as follows:

Definition 2. ***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., the topic-word matrix Φ, the region-topic matrix Ψ, and the Gaussian distribution $\mathcal{N}(\mu_r, \Sigma_r)$ of each region, on the data distributed to the workers in parallel.*

5 PGeoTopic

In this section, we first provide an overview of our proposed solution (Section 5.1). Then, we present the key techniques of the proposed solution (Section 5.2 - 5.4). Finally, we give a summary of the whole training process (Section 5.5).

5.1 PGeoTopic Overview

To the best of our knowledge, there is no existing distributed training solution for geographical topic models. We start with a straightforward method with only data parallelism to motivate the design objectives of PGeoTopic. A straightforward method is to evenly distribute the data randomly to each worker. In each iteration, each worker fetches the parameters from the parameter servers to its local memory and use Eq. (1) to sample the topic and region for each word token in the documents assigned to the worker. When parameters cannot 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. At the end of each iteration, the updated parameters are pushed back to the parameter servers.

The shortcomings of this method 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, a large model may not fit into memory. This method has to frequently swap the parameters between memory and hard disk, incurring expensive I/O cost. Thirdly, access and synchronization of the parameters within iterations incurs heavy network communication cost.

Design Objectives. The motivation of this work is to address these issues. Specifically, we propose the following design objectives:

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

Based on the three design objectives, we propose a distributed solution, called PGeoTopic. PGeoTopic comprises

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

three novel techniques – A new distributed training algorithm (Section 5.2), a model parallelism (Section 5.3) and a parameter localization mechanism (Section 5.4).

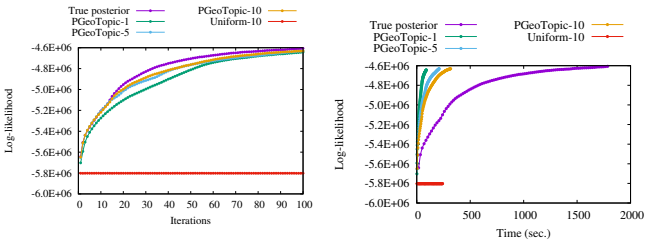
5.2 The Proposed Distributed Training Algorithm

Consider sampling the topic and region for a word token using Eq. (1). Even though we can only fetch the topic vector of one word token each time from the word-topic matrix, the whole topic-region matrix is needed before sampling a topic and region for every 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 interdependency of the two matrices** and make use of them independently. The high-level idea is to divide the word tokens into two sets, and for one set we use the word-topic matrix and the region parameters (e.g., $f(l_d|\mu_r, \Sigma_r) \cdot (n_r + \alpha_r)$ in Eq. (1)) to sample topics and regions while using the topic-region matrix to sample topics and regions for the other set.

To realize the high-level idea, we propose a new training algorithm that uses Metropolis-Hastings [38] as building block. Metropolis-Hastings (MH) algorithm is often used to draw samples from distributions that are hard to sample from. The idea is to sample from a **proposal** distribution and test whether each 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. It often takes several sampling steps (**burn-in steps**) for MH to generate samples similar to the original distribution.

In our case, the original distribution is the posterior in Eq. (1), and we aim to derive a proposal distribution that supports independent access to the two parameter matrices.
Challenges. Designing a proper proposal distribution is usually difficult. The proposal distribution should be able to draw similar samples as does Eq. (1), and be able to converge to similar likelihood within similar number of iterations as does Eq. (1). Meanwhile, the sampling efficiency should be as high as possible, and support independent access to the two matrices. Figure 3 shows the training effectiveness and efficiency of some example proposals. Uniform distribution with 10 burn-in 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, sampling from true posterior (True posterior) needs to access both matrices and is often time-consuming. Our goal is to construct a proposal like PGeoTopic in Figure 3 that achieves effective and efficient training and supports independent access to the word-topic and topic-region matrices.
Observations. To address the aforementioned challenges, we develop our training algorithm based on the following three observations.

- **Observation 1**: If we only consider the word-topic and location-region correlations, sampling a topic from the word-topic distribution and a region from the posterior of Gaussian mixture are independent, i.e., $p(z_{di} = k, r_{di} = r|D) \approx p(z_{di} = k|\mathbf{w}_d) \cdot p(r_{di} = r|\mathbf{l}_d)$.
- **Observation 2**: If we only consider the topic-region correlation, sampling a topic-region pair with probability proportional to m_{kr} is similar to sampling from a data-independent joint distribution $p(z_{di} = k, r_{di} = r)$.
- **Observation 3**: The proposal distribution should preserve all the correlations in Eq. (1) to draw similar samples as does Eq. (1).



(a) Log-Likelihood v.s. Iterations (b) Log-Likelihood v.s. Time

Fig. 3: Training effectiveness and efficiency on a random sample of 10K geo-tagged tweets, where N of $PGeoTopic-N$ is the number of burn-in steps. Both numbers of topics and regions are set at 10.

To tackle the aforementioned challenges, we propose two proposal distributions, namely independent proposal and joint proposal, respectively, and apply a method to combine the two proposals to preserve all the correlations in Eq. (1) in order to generate similar samples to Eq. (1).
Independent proposal. We define the independent proposal as:

$$q_{indep}(z_{di} = k, r_{di} = r) \propto \frac{n_{kw} + \beta_{kw}}{n_k + \bar{\beta}} \cdot f(l_d|\mu_r, \Sigma_r) \cdot (n_r + \alpha_r), \tag{2}$$

where $n_k = \sum_{w'} n_{kw'}$ and $\bar{\beta} = \sum_{w'} \beta_{kw}$.

In this proposal, we sample a topic with probability proportional to $\frac{n_{kw}+\beta_w}{n_k+\bar{\beta}}$ and a region from $f(l_d|\mu_r, \Sigma_r) \cdot (n_r + \alpha_r)$ independently. For simplicity, we use $p(k, r)$ to denote $p(z_{di} = k, r_{di} = r|\Psi^{-di}, \Phi^{-di}, D)$ (Eq. (1)) and $q_{indep}(k, r)$ to denote $q_{indep}(z_{di} = k, r_{di} = r)$. The acceptance ratio of the samples drawn from the proposal is then computed as:

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

where t is the t-th sample, and $t-1$ represents the previous sample.

In the independent proposal, the term $\frac{n_{kw}+\beta_w}{n_k+\bar{\beta}}$ ensures similar topic-word correlations as does the true posterior, while $f(l_d|\mu_r, \Sigma_r) \cdot (n_r + \alpha_r)$ 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 topic-region correlation. It is defined as:

$$q_{joint}(z_{di} = k, r_{di} = r) \propto (m_{kr} + \gamma_{kr}) \cdot f(l_d|\mu_r, \Sigma_r) \cdot (n_r + \alpha_r), \tag{3}$$

with acceptance ratio 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 the topic-region matrix and the region parameters. It is used to make sure the learned topics and regions are correlated. We keep the Gaussian mixture term $f(l_d|\mu_r, \Sigma_r) \cdot (n_r + \alpha_r)$ to support our proposed model parallelism (Section 5.3).

Combining the two proposals. We combine the two proposals using the cyclic method [39]. Specifically, after applying the independent proposal to a word token, 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 equals to the multiplication of the densities of two proposals [39]. In our case, the multiplication of the two proposals results in the similar form of Eq. (1).

Note that we need to access the topic-region matrix to compute the density $p(z_{di} = k_t, r_{di} = r_t)$ in the acceptance ratio of the independent proposal. The same problem exists in the joint proposal. To allow using the two matrices independently for drawing samples, we delay the computation of the acceptance ratio to the end of each iteration as does WrapLDA [26]. In other words, we first draw a sequence of s samples (z_{di}, r_{di}) using the two proposals for each word token, where s is the number of burn-in steps. Given that all the samples are drawn, we then compute the true posterior distribution $p(z_{di} = k_t, r_{di} = r_t)$ and the acceptance ratio by accessing the corresponding parameters, and accept/reject the samples in the sample sequence of each word token. We present the details in Section 5.3.

5.3 Model Parallelism

Based on the proposed training algorithm, we design a model parallelism that supports the parallelism of training and network communication, and reduce the memory requirement of worker machines. 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 n_r and topic counts n_k, are of small size. Our high level idea is to partition the two matrices into slices, and each time a worker machine only fetches one slice for training. When a worker is sampling with one slice, it prefetches the next slice to parallel the training computation and communication. Figure 4 illustrates the proposed model parallelism.

Parallelism of Word-Topic Matrix. Before training, we allocate the independent (white) and joint (grey) proposals alternatively to the word tokens in the data of a worker. As a result, sampling for the words in white only needs to access the word-topic matrix while sampling for the words in grey only needs to access the topic-region matrix. Then, we partition the word-topic matrix by words into word slices (blue rectangle in Figure 4) and the topic-region matrix by regions into region slices (in red rectangle in Figure 4). Given a training document and a word slice on a worker, we only sample the topics and regions for the word tokens of the document that are contained in the word slice and allocated to the independent proposal.

Parallelism of Topic-Region Matrix. For the topic-region matrix, the case is different because the whole topic-region matrix may be accessed for each single word token in the joint proposal. As such, we propose a two-step sampling scheme. The idea is to first sample regions for each word token. Given a document and a region slice, we then only sample topics for the word tokens that are assigned to regions in the region slice. Specifically, the joint proposal can be viewed as a mixture model, i.e., the probability of sampling a topic k from it is $p(z_{di} = k) = \sum_r p(z_{di} = k, r_{di} = r)p(r_{di} = r)$, where $p(r_{di} = r) \propto f(g_d|\mu_r, \Sigma_r) \cdot (n_r + \alpha_r)$. Our two-step sampling scheme follows the sampling routine of a mixture model. In the first step, we sample a region for a word token based on $f(l_d|\mu_r, \Sigma_r) \cdot (n_r + \alpha_r)$; 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 $m_{kr} + \gamma_{kr}$.

For example in Figure 4, given document d_4 and region slice $r_1 \sim r_2$, we first sample the regions of the word tokens in d_4 (e.g., r_2 and r_3). Then, we sample the topics for the word tokens assigned to region r_2 (in red color). After the region slice $r_3 \sim r_4$ is fetched, we sample the topics for the word tokens assigned to region r_3 (in black color). In each training iteration, a worker fetches the two matrices slice by slice from the parameter servers. After using a word slice or a region slice, the updates are pushed to the parameter servers and the local copy of the slice can be removed from memory. In addition, when sampling with a parameter slice we can prefetch the next parameter slice at the same time to hide the network communication time.

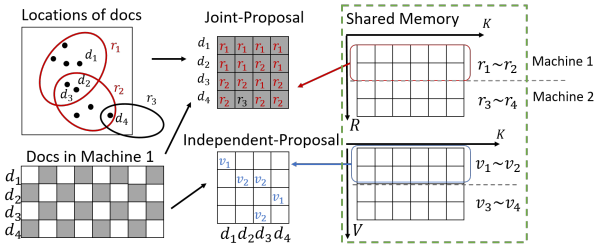


Fig. 4: An example of model parallelism.

5.4 Parameter Localization

To reduce the communication cost, we first use spatial partitioning techniques to ensure documents in each worker are spatially close, and only relate to part of the regions. Without loss of generality, we use k-d tree [40] to partition the data[3]. Then, we propose two ideas for parameter localization – (1) on-demand fetching of topic-region matrix and (2) parameter allocation to related worker.

5.4.1 On-demand Fetching of Topic-Region Matrix

As we partition the data according to the spatial information (e.g., using k-d tree), the documents in a worker are spatially close. When we apply the two-step sampling scheme in Section 5.3, regions that are far from the document are less likely to be sampled in the first step, because the Gaussian

3. We leave the comparison among spatial partitioning techniques as future work.

distribution has exponential decrement of density w.r.t. the distance. As a result, only a small number of regions are close to the documents in a particular worker machine, and the worker only needs to access relevant region slices (contain regions sampled from the first step of the two-step sampling scheme) without iterating all the region slices. Benefit from spatial partitioning, this on-demand fetching manner largely reduce the chance of accessing the topic-region parameters stored in the remote parameter servers.

5.4.2 Parameter Allocation to Related Worker

The parameter servers store the 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 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 slices from the parameter servers only needs access local memory. We achieve this by initializing the Gaussian mixture parameters of the regions with the data in the same machine. For example in Figure 4, 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 region slices for r_1 and r_2.

5.5 Summary of the Training Process

Algorithm 1 shows the sampling process in a worker machine. The data allocated to the worker is split into data blocks that can fit into memory and each time we read a data block from disk. To process a data block, we first allocate one of the two proposals (independent or joint) to each word token alternatively as mentioned in Section 5.2 (Line 5). After that, we sample a sequence of s (burn-in steps) pairs of topic-region for Metropolis-Hastings. For the t-th sample of region-topic pair, we use $R_{di}[t]$ and $K_{di}[t]$ to denote its region and topic, respectively. Specifically, we draw s regions from the posterior of the Gaussian mixture, i.e., $f(l_d|\mu_r, \Sigma_r) \cdot (n_r + \alpha_r)$, and store the samples in R_{di} and the corresponding density in p_{di} and q_{di} (Lines 6 - 9). With all the regions sampled, we next sample topics for each word token. We fetch the topic-region parameters slice by slice and sample s topics for each word token allocated to the joint proposal (Lines 11 - 18). During the sampling process, we also update and store the density (Lines 17 - 18). Similarly, we fetch the word-topic parameters slice by slice and sample s topics for each word token allocated to independent proposal (Lines 19 - 27). Because we have sampled topics and regions for all word tokens allocated to the joint proposal, we can now update the true posterior density $p(z_{di} = k_t, r_{di} = r_t)$ of all word tokens allocated to both proposals at the same time when we are working on the word slices (Line 27). To this end, we have computed and stored the true posterior density for all word tokens allocated to the joint proposal. We then update the true posterior density for word tokens allocated to the independent proposal by accessing necessary topic-region parameters[4] (Lines 28 - 30). Finally, we compute the acceptance ratio and accept/reject the samples for each word token (Lines 32 - 35). The resulting region assignment for word tokens in a document forms an empirical region distribution $p(r_d|D) = \frac{c_r^d}{N_d}$, where c_r^d is the number of words assigned to region r in the document d, and N_d is the total number of word tokens in d. Directly sampling from such distribution takes $O(R)$ time. As such, we sample the region of the document by randomly selecting from one of its word's region assignment $r_{di'}$ (Lines 36 - 38). Because there are c_r^d out of N_d words assigned to region r, this sampling process is equivalent to sample from the empirical region distribution. We terminate the sampling process when the model converges, i.e., the increment of likelihood is less than a percentage threshold (e.g., 0.1%).

Note that in Algorithm 1, each worker reads and samples data blocks as well as fetches parameter slices from parameter servers during sampling. We can further reduce the training time by overlapping I/O, network and computation, i.e., we prefetch the next data block and the next parameter slice while sampling with current data block. Specifically, 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, while the model thread keeps retrieving the next parameter slice from network.

5.6 Extension of PGeoTopic

Geographical topic models have many variants [1], [4], [10], [12], [20], [21] in which other variables such as user and time are considered. Suppose we have a set of additional variables $\mathbf{X}$ correlate to topics and/or regions (e.g., time), the Gibbs Sampler for a model with additional variables $\mathbf{X}$ is often in the following form:

$$\begin{aligned} &p(z_{di} = k, r_{di} = r, \mathbf{X}|\Psi^{-di}, \Phi^{-di}, D) \\ \propto &p(z_{di} = k, r_{di} = r|\Psi^{-di}, \Phi^{-di}, D) \cdot g^{-di}(\mathbf{X}|k, r), \end{aligned} \tag{4}$$

where $g^{-di}(\mathbf{X}|k, r)$ is the posterior predictive distribution excluding the current word token. Our solution can be applied to these variants with minor modification. Consider the variables $\mathbf{X}_z \subseteq \mathbf{X}$ that are only related to topic, and $\mathbf{X}_r \subseteq \mathbf{X}$ that are only related to region, we extend the independent proposal as:

$$\begin{aligned} q'_{indep}(z_{di} = k, r_{di} = r) \propto &q_{indep}(z_{di} = k, r_{di} = r) \cdot \\ &g^{-di}(\mathbf{X}_z|k) \cdot g^{-di}(\mathbf{X}_r|r), \end{aligned} \tag{5}$$

such that the parameters in the independent proposal are related to either topic or region. The parameters related to topics and regions can then be partitioned into slices by words and regions, respectively, for parallelism. For variables $\mathbf{X}_{zr} \subseteq \mathbf{X}$ that are related to both topic and region, we extend the joint proposal as:

$$\begin{aligned} q'_{joint}(z_{di} = k, r_{di} = r) \propto &q_{joint}(z_{di} = k, r_{di} = r) \cdot \\ &g^{-di}(\mathbf{X}_{zr}|k, r), \end{aligned} \tag{6}$$

4. We can buffer these necessary parameters in the local memory when we work on the region slices to avoid fetching these parameters via network.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.2989142, IEEE Transactions on Knowledge and Data Engineering

Algorithm 1: Training process in a worker

while *The model has not converged* **do**
  foreach *data block* D_b *in the worker* **do**
    $Q \leftarrow \emptyset$;
    foreach d *in* D_b *and* i *in* $\mathbf{w}_d$ **do**
      Allocate the two proposals to w_{di} alternatively;
      for *t=1 to s* **do**
        $R_{di}[t] \leftarrow$ Sample a region $r_t \propto f(l_d|\mu_r, \Sigma_r) \cdot (n_r + \alpha_r)$;
        $p_{di}[t] \leftarrow f(l_d|\mu_{r_t}, \Sigma_{r_t}) \cdot (n_{r_t}^{-d} + \alpha_{r_t})$;
        $q_{di}[t] \leftarrow f(l_d|\mu_{r_t}, \Sigma_{r_t}) \cdot (n_{r_t} + \alpha_{r_t})$;
      $Q \leftarrow Q \cup \{$ unique regions from $R_{di}\}$;
    while *Has next region slice* **do**
      Fetch next region slice that has regions in Q;
      foreach d, i *where* r_{di} *is in the region slice* **do**
        if *Joint proposal is allocated* **then**
          for *t=1 to s* **do**
            $Z_{di} \leftarrow$ Sample a topic $k_t \propto m_{kR_{di}[t]} + \gamma_{kR_{di}[t]}$;
            $p_{di}[t] \leftarrow p_{di}[t] \cdot (m^{-di}_{k_t R_{di}[t]} + \gamma_{k_t R_{di}[t]})$;
            $q_{di}[t] \leftarrow q_{di}[t] \cdot (m_{k_t R_{di}[t]} + \gamma_{k_t R_{di}[t]})$;
    while *Has next word slice* **do**
      Fetch next word slice;
      foreach d, i *where* w_{di} *in the word slice* **do**
        if *Independent proposal is allocated* **then**
          for *t=1 to s* **do**
            $Z_{di}[t] \leftarrow$ Sample a topic $k_t \propto \frac{n_{kw_{di}} + \beta_{kw_{di}}}{n_k + \bar{\beta}}$;
            $q_{di}[t] \leftarrow q_{di}[t] \cdot \frac{n_{k_t w_{di}} + \beta_{k_t w_{di}}}{n_{k_t} + \bar{\beta}}$;
        for *t=1 to s* **do**
          $p_{di}[t] \leftarrow p_{di}[t] \cdot \frac{n^{-di}_{k_t w_{di}} + \beta_{k_t w_{di}}}{n^{-di}_{k_t} + \bar{\beta}}$;
    foreach w_{di} *allocated to independent proposal* **do**
      for *t=1 to s* **do**
        $p_{di}[t] \leftarrow p_{di}[t] \cdot (m^{-di}_{Z_{di}[t] R_{di}[t]} + \gamma_{Z_{di}[t] R_{di}[t]})$;
    foreach d *in* D_b *and* i *in* $\mathbf{w}_d$ **do**
      $z_{di}, r_{di} \leftarrow Z_{di}[1], R_{di}[1]$;
      for *t=2 to s* **do**
        if *Random(0,1) <* *AcceptanceRatio*($p_{di}[t-1]$, $p_{di}[t]$, $q_{di}[t-1]$, $q_{di}[t]$) **then**
          $z_{di}, r_{di} \leftarrow Z_{di}[t], R_{di}[t]$;
    foreach d *in* D_b **do**
      $i' \leftarrow$ random(1,$|\mathbf{w}_d|$);
      $r_d \leftarrow r_{di'}$;

and parallelize the parameters related to $\mathbf{X}_{zr}$ in the same way as the topic-region matrix. Then, we can apply Algorithm 1 on the extended proposals.

6 Experiments

6.1 Experimental Setup

Datasets. We conduct experiments on three datasets, namely *Twitter*, *Yelp* and *Web*. The Twitter dataset is collected from New York city users during 2015, which contains more than 136 million geo-tagged documents. The Yelp dataset[5] contains 4.40 million user reviews on points of interest in four states (Nevada, Utah, California and Arizona) of USA. We use the geo-location of the points of interest as the location of the reviews. The Web dataset is collected from the Common Crawl data[6], which contains 22 millions web pages. Because the web pages do not have geo-tags, we randomly assign the web pages to a POI dataset SimpleGeo's Places[7] with 12 millions locations in United States. Note that we only use the Web dataset for evaluating the training efficiency. Table 3 reports the statistics of both datasets.

TABLE 3: Statistics of the datasets

	Twitter	Yelp	Web
#docs	136.69 M	4.40 M	22.04 M
#tokens	755.80 M	234.47 M	1.30 B
vocabulary	316,890	153,621	162,694
latitude	[40.4, 40.9]	[33.2, 41.2]	[18.3, 71.3]
longitude	[-74.2, -73.7]	[-115.4, -111.6]	[-168.0, -64.9]

Distributed training solutions. We compare the proposed solution (PGeoTopic) with the following baseline solutions:

- **Baseline**: The straightforward solution in Section 5.1.
- **Baseline+M**: We extend Baseline by applying the model parallelism of LightLDA [25], [41] on the word-topic matrix.
- **PGeoTopic-O**: PGeoTopic without the on-demand parameter fetching method (Section 5.4.1).
- **PGeoTopic-L**: PGeoTopic without allocating the parameters to related workers (Section 5.4.2).
- **PGeoTopic-P**: PGeoTopic without overlapping I/O, computation and network communication (Section 5.5).
- **PGeoTopic-S**: PGeoTopic that distributes data to worker machines randomly, but not based on spatial partitioning.

We set both the number of topics and the number of regions at 10,000 if not specified. 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. Our solution can be built based on most of the existing parameter servers. In this paper, we implement the proposed solution on top of an

5. https://www.yelp.com/dataset
6. https://aws.amazon.com/public-datasets/common-crawl/
7. https://freegisdata.rtwilson.com/

open-sourced[8] stale-synchronized parameter server called Bösen [30]. We setup the experiments on an Amazon EC2 cluster with 20 machines, each of which has 27 computation units, 61 GB RAM and 1 Gbps network. The default number of machines used for training is 20 if not specified.

Objectives. (1) We evaluate the scalability of PGeoTopic with model size, number of machines and data size (Section 6.2); (2) We evaluate the time used for computation versus network communication to show the usefulness of overlapping the computation and network communication (Section 6.3); (3) We evaluate the memory cost of worker machines and the network communication cost (Section 6.4 and Section 6.5); (4) We also evaluate the model quality via two applications (Section 6.6) and present a case study (Section 6.7).

6.2 Scalability

We first show that PGeoTopic is able to handle a big model with 100,000 topics and 100,000 regions. We run Baseline, Baseline+M and PGeoTopic on 20 machines on the Twitter dataset. We observe that Baseline and Baseline+M cannot finish one iteration of training within one week. PGeoTopic finishes training within 20 hours. This is because PGeoTopic improves parallelism and reduces network communications. In contrast, the two baseline methods do not consider the parallelism on the topic-region matrix, thus incurring high network communication.

We follow Yuan et al. [25] to use average throughput instead of overall training time in the remaining scalability experiments because it is impractical to run the baseline methods until they converge. The average throughput is measured by the average number of word tokens sampled per second. It is roughly inversely proportional to the training time because all methods converge in similar numbers of iterations as observed from the results on small datasets. We run each baseline method for 10 hours and count the number of total tokens sampled for computing the average throughput. For PGeoTopic, we run util it converges.

Scalability with model size. We first 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 topics. 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}. We report the average throughput of all methods in Figure 5 and Figure 6.

In all datasets, PGeoTopic outperforms the two baseline methods by several orders of magnitude. The throughput of the two baselines degrades fast as the number of topics/regions increases. Compared to the baselines, PGeoTopic is less sensitive to the number of topics/regions. When the number of topics/regions increases, the throughput of PGeoTopic slightly degrades. This is because when the model is large, it is more difficult to overlap the network communication cost with the computation.

Scalability with number of machines. We run PGeoTopic, Baseline and Baseline+M on 12, 16 and 20 machines, respectively, and report the average throughput in Figure 7. Both baseline methods have extremely low throughput, i.e., less than 0.2 tokens per second on all the datasets. The proposed solution scales roughly linearly to the growth of number of machines in Twitter and Web. In Yelp dataset, the performance of PGeoTopic does not increase significantly when the number of machines increase from 16 to 20. This may because the Yelp dataset is smaller than the other two, and thus each worker machine has less computation cost to overlap with the network communication. Overall, our solution is efficient and scalable w.r.t. the number of machines, because PGeoTopic overlaps the computation with network communication, and reduces the network communication cost.

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. The average throughput is reported in Figure 8. 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 largely overlap the time for network communication. In contrast, the two baselines do not display the property.

6.3 Computation v.s. Network and I/O

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 9. Both the number of topics and the number of regions are set at 10,000.

Figure 9 shows that PGeoTopic-P is 1–3 times more expensive than PGeoTopic for communication cost while they take similar time for computation. In addition, we observe that PGeoTopic-P takes 23.5 seconds on average for I/O per iteration while PGeoTopic only takes 2 seconds on average. The reason is that PGeoTopic-P does not overlap computation with network communication and I/O. Our experimental results demonstrate the benefits of overlapping computation with network communication and I/O.

6.4 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.

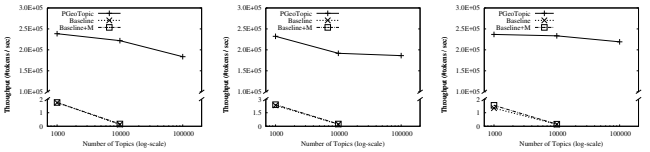
TABLE 4: Memory usage of PGeoTopic in gigabytes

# 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. The worker memory usage remains stable irrespective of the number of machines. This is because PGeoTopic only needs to load one model slice into memory each time. This experiment indicates that PGeoTopic is applicable to machines with limited memory.

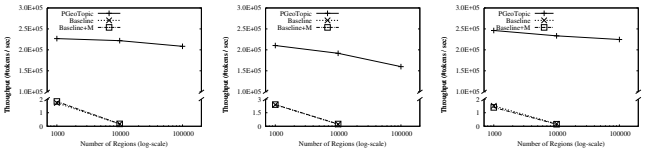
8. https://github.com/sailing-pmls/jbosen

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.2989142, IEEE Transactions on Knowledge and Data Engineering



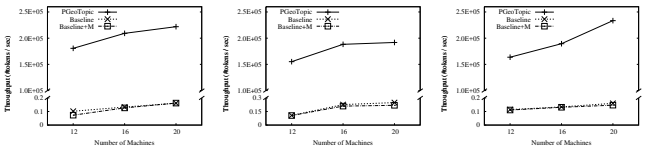
(a) Twitter (b) Yelp (c) Web

Fig. 5: Varying number of topics



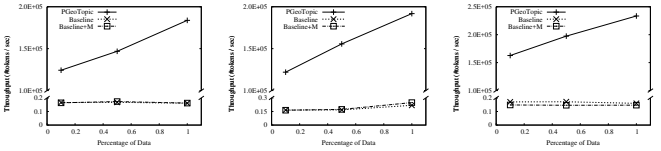
(a) Twitter (b) Yelp (c) Web

Fig. 6: Varying number of regions



(a) Twitter (b) Yelp (c) Web

Fig. 7: Varying number of machines



(a) Twitter (b) Yelp (c) Web

Fig. 8: Varying data size

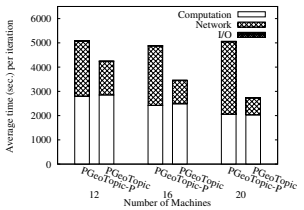


Fig. 9: Training time breakdown - computation v.s. network communication and I/O per iteration.

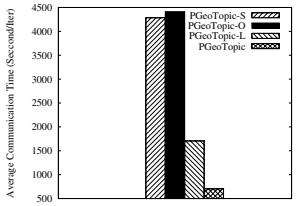


Fig. 10: Average network communication time (in sec.) per iteration.

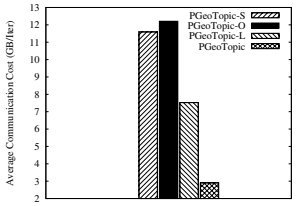


Fig. 11: Average size of the topic-region parameters (in GB) transmitted via network per iteration.

6.5 Network Communication Cost

We evaluate the effect of the parameter localization of PGeoTopic in reducing the communication cost. Specifically, we compare PGeoTopic with three variants — (1) PGeoTopic-O that disables the on-demand parameter fetching; (2) PGeoTopic-L that assigns the region slices to the machines randomly without allocating them to related workers; (3) PGeoTopic-S that randomly allocates the data to workers without spatial partitioning. We run all the methods on 20 machines, and report the average network communication time and the size of the topic-region parameters transmitted via network in each iteration in Figure 10 and Figure 11, 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. PGeoTopic performs best and reduces the network communication cost significantly. This is because PGeoTopic achieves parameter localization by allocating the parameters to related workers.

6.6 Model Quality

We investigate the model quality on two downstream applications of geographical topic models – (1) Predicting new documents [18]; and (2) Predicting location of documents without geo-tags [10]. In new document prediction, we apply the learned models to predict new documents and we use perplexity (the lower the better) to measure how likely the models can predict new documents. For location prediction, we take the text of a tweet/review as input and predict the coordinates where the tweet/review was posted. We use distance errors in kilometers to evaluate how well the models can predict locations given the words. We train both Baseline and PGeoTopic until the models converge. Because Baseline cannot train large models on large datasets, we randomly sample 10% tweets from the Twitter dataset. For Yelp dataset, we use the whole dataset. For both datasets, we hold off 10% of the data for testing and use the remaining 90% data for training.

Model quality w.r.t. number of regions. We learn 100 topics and {20, 50, 100} regions on 8 machines and report the results in Table 5 and Table 6. We observe that both perplexity and location prediction error decrease as the number of regions increases in both datasets. This is because we can learn finer-grained topical regions, which contains local semantics (i.e., topic distribution) and more accurate location information (i.e., mean and variance). For document prediction, PGeoTopic has only 5%-6.5% higher perplexity than Baseline. For location prediction, PGeoTopic increases the error by 1% in the worst case. Overall, PGeoTopic achieves similar performance to Baseline in both tasks.

TABLE 5: Document prediction (Perplexity)

	# Regions	20	50	100
Twittter	Baseline	1721.25	1245.31	1124.41
	PGeoTopic	1614.24	1327.30	1180.51
Yelp	Baseline	809.65	804.49	420.15
	PGeoTopic	810.65	765.83	401.67

TABLE 6: Location prediction (Error in km)

	# Regions	20	50	100
Twittter	Baseline	28.55	17.66	10.14
	PGeoTopic	26.21	17.87	10.18
Yelp	Baseline	12.83	12.80	5.11
	PGeoTopic	12.34	11.78	5.36

Model quality w.r.t. number of machines. We learn 100 topics and 100 regions on {8, 12, 16} machines and report the results in Table 7 and Table 8. Both Baseline and PGeoTopic perform similarly with different number of worker machines on both datasets. The difference of perplexities PGeoTopic among different number of machines is around 1%-2%, and the distance errors change within 2 kilometers on both datasets. This experiment shows that PGeoTopic retains the model quality by increasing the number of worker machines, and thus is effective to train on larger computer clusters.

Summary. Combining the previous scalability experiments, we conclude that PGeoTopic can efficiently train big geographical topic models on large datasets while preserving the model quality.

TABLE 7: Document prediction (Perplexity)

	# Machines	8	12	16
Twittter	Baseline	1124.41	1143.67	1109.81
	PGeoTopic	1180.51	1161.05	1191.24
Yelp	Baseline	420.15	425.00	415.56
	PGeoTopic	401.67	403.13	404.78

TABLE 8: Location prediction (Error in km)

	# Machines	8	12	16
Twittter	Baseline	10.14	10.52	10.10
	PGeoTopic	10.18	11.75	12.02
Yelp	Baseline	5.19	5.23	5.11
	PGeoTopic	5.36	5.18	4.90

6.7 Case Study

We show some example latent regions and topics. In particular, we select 100 most informative regions from the 10,000 regions learned from Twitter dataset, ranked by the entropy w.r.t. the topic distribution of each region. Regions with high entropy only mention particular topics rather than covering all topics. We randomly select two regions from the map and two other regions near "Newark Liberty International Airport". We plot the four regions in Figure 12. The top two topics and their representative words (i.e., words with high probability) for each region are listed in Table 9.

Observe from Figure 12a and Figure 12b, the university area covers topics such as class and campus life, while the topics in beach area focus on entertainment (e.g., wearing, eating, and drinking). In addition, Figure 12c and Figure 12d shows that even in the same airport, the model discovered different topical-regions such as gates and lounges. Topics in lounges are mainly about things to do for time killing and flight delays, while in the gate area, people often talk about arrival/departure and carrier information.

(a) University (b) Beach

(c) Airport - lounges (d) Airport - gate

Fig. 12: Example regions learned on the Twitter dataset. The ovals are the contour lines at confidence level $p = 0.9$. The pins point out the centers of the regions.

TABLE 9: Topics for each region

Region	Topic	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 (lounges)	958	urge, phone, @mangomeagan, Playing, fm
	7955	delayed, #Weather, fan, engaged, destined
Airport (gate)	9321	Uber, ending,#z100MendesMadness,ready, fam
	4428	airport, let, carrier, #cathedral, hashtag

7 Conclusions

In this paper, we propose PGeoTopic, the first distributed training solution for geographical topic models. PGeoTopic increases the parallelism and reduce the memory requirements with a novel training algorithm and model parallelism. It also reduces network communication by localizing parameters to related worker machines. Our experimental results demonstrate that the proposed solution is scalable w.r.t. number of machines, data size and model size, and outperforms baseline solutions by orders of magnitude.

Acknowledgments. This work was supported in part by a MOE Tier-2 grant MOE2016-T2-1-137, a MOE Tier-1 grant RG31/17, and NSFC under the grant 61772537.

REFERENCES

[1] H. Yin, Y. Sun, B. Cui, Z. Hu, and L. Chen, "LCARS: a location-content-aware recommender system," in *KDD*, 2013, pp. 221–229.

[2] B. Liu, Y. Fu, Z. Yao, and H. Xiong, "Learning geographical preferences for point-of-interest recommendation," in *KDD*, 2013, pp. 1043–1051.

[3] B. Liu, H. Xiong, S. Papadimitriou, Y. Fu, and Z. Yao, "A general geographical probabilistic factor model for point of interest recommendation," *TKDE*, vol. 27, no. 5, pp. 1167–1179, 2015.

[4] Y. Liu, M. Ester, B. Hu, and D. W. Cheung, "Spatio-temporal topic models for check-in data," in *ICDM*, 2015, pp. 889–894.

[5] W. Wang, H. Yin, L. Chen, Y. Sun, S. W. Sadiq, and X. Zhou, "Geosage: A geographical sparse additive generative model for spatial item recommendation," in *KDD*, 2015, pp. 1255–1264.

[6] H. Yin, X. Zhou, B. Cui, H. Wang, K. Zheng, and N. Q. V. Hung, "Adapting to user interest drift for POI recommendation," *TKDE*, vol. 28, no. 10, pp. 2566–2581, 2016.

[7] H. Yin, B. Cui, X. Zhou, W. Wang, Z. Huang, and S. W. Sadiq, "Joint modeling of user check-in behaviors for real-time point-of-interest recommendation," *TOIS*, vol. 35, no. 2, pp. 11:1–11:44, 2016.

[8] W. Wang, H. Yin, S. W. Sadiq, L. Chen, M. Xie, and X. Zhou, "SPORE: A sequential personalized spatial item recommender system," in *ICDE*, 2016, pp. 954–965.

[9] H. Yin, W. Wang, H. Wang, L. Chen, and X. Zhou, "Spatial-aware hierarchical collaborative deep learning for POI recommendation," *TKDE*, vol. 29, no. 11, pp. 2537–2551, 2017.

[10] Q. Yuan, G. Cong, K. Zhao, Z. Ma, and A. Sun, "Who, where, when, and what: A nonparametric bayesian approach to context-aware recommendation and search for twitter users," *ACM Trans. Inf. Syst.*, vol. 33, no. 1, pp. 2:1–2:33, 2015.

[11] L. Zhao, F. Chen, C.-T. Lu, and N. Ramakrishnan, "Spatiotemporal event forecasting in social media," in *SDM*, vol. 15, 2015, pp. 963–971.

[12] J. Guo and Z. Gong, "A nonparametric model for event discovery in the geospatial-temporal space," in *CIKM*, 2016, pp. 499–508.

[13] L. Hong, A. Ahmed, S. Gurumurthy, A. J. Smola, and K. Tsioutsiouliklis, "Discovering geographical topics in the twitter stream," in *WWW*, 2012, p. 769.

[14] K. Zhao, G. Cong, and A. Sun, "Annotating points of interest with geo-tagged tweets," in *CIKM*, 2016, pp. 417–426.

[15] Q. Liu, Y. Ge, Z. Li, E. Chen, and H. Xiong, "Personalized travel package recommendation," in *ICDM*, 2011, pp. 407–416.

[16] Q. Liu, E. Chen, H. Xiong, Y. Ge, Z. Li, and X. Wu, "A cocktail approach for travel package recommendation," *TKDE*, vol. 26, no. 2, pp. 278–293, 2014.

[17] S. Sizov, "Geofolk: Latent spatial semantics in web 2.0 social media," in *WSDM*, 2010, pp. 281–290.

[18] Z. Yin, L. Cao, J. Han, C. Zhai, and T. Huang, "Geographical topic discovery and comparison," in *WWW*, 2011, pp. 247–256.

[19] A. Ahmed, L. Hong, and A. J. Smola, "Hierarchical geographical modeling of user locations from social media posts," in *WWW*, 2013, pp. 25–36.

[20] L. Hong, A. Ahmed, S. Gurumurthy, A. J. Smola, and K. Tsioutsiouliklis, "Discovering geographical topics in the twitter stream," in *WWW*, 2012, pp. 769–778.

[21] B. Hu, M. Jamali, and M. Ester, "Spatio-temporal topic modeling in mobile social media for location recommendation," in *ICDM*, 2013, pp. 1073–1078.

[22] A. Ahmed, M. Aly, J. Gonzalez, S. Narayanamurthy, and A. J. Smola, "Scalable inference in latent variable models," in *WSDM*, 2012, pp. 123–132.

[23] Z. Liu, Y. Zhang, E. Y. Chang, and M. Sun, "Plda+: Parallel latent dirichlet allocation with data placement and pipeline processing," *TIST*, vol. 2, no. 3, pp. 26:1–26:18, May 2011.

[24] Y. Wang, X. Zhao, Z. Sun, H. Yan, L. Wang, Z. Jin, L. Wang, Y. Gao, J. Zeng, Q. Yang *et al.*, "Towards topic modeling for big data," *arXiv preprint arXiv:1405.4402*, 2014.

[25] J. Yuan, F. Gao, Q. Ho, W. Dai, J. Wei, X. Zheng, E. P. Xing, T.-Y. Liu, and W.-Y. Ma, "Lightlda: Big topic models on modest computer clusters," in *WWW*, 2015, pp. 1351–1361.

[26] J. Chen, K. Li, J. Zhu, and W. Chen, "Warplda: a cache efficient o (1) algorithm for latent dirichlet allocation," *PVLDB*, vol. 9, no. 10, pp. 744–755, 2016.

[27] H.-F. Yu, C.-J. Hsieh, H. Yun, S. Vishwanathan, and I. S. Dhillon, "A scalable asynchronous distributed algorithm for topic modeling," in *WWW*, 2015, pp. 1340–1350.

[28] D. Newman, A. Asuncion, P. Smyth, and M. Welling, "Distributed algorithms for topic models," *Journal of Machine Learning Research*, vol. 10, no. Aug, pp. 1801–1828, 2009.

[29] C. Zhang and C. Ré, "Towards high-throughput gibbs sampling at scale: A study across storage managers," in *SIGMOD*, 2013, pp. 397–408.

[30] Q. Ho, J. Cipar, H. Cui, S. Lee, J. K. Kim, P. B. Gibbons, G. A. Gibson, G. Ganger, and E. P. Xing, "More effective distributed ml via a stale synchronous parallel parameter server," in *NIPS*, 2013, pp. 1223–1231.

[31] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, "Scaling distributed machine learning with the parameter server." in *OSDI*, vol. 14, 2014, pp. 583–598.

[32] A. Smola and S. Narayanamurthy, "An architecture for parallel topic models," *PVLDB*, vol. 3, no. 1-2, pp. 703–710, 2010.

[33] S. Sizov, "Latent Geospatial Semantics of Social Media," *TIST*, vol. 3, no. 4, pp. 1–20, 2012.

[34] C. P. Robert, V. Elvira, N. Tawn, and C. Wu, "Accelerating mcmc algorithms," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 10, no. 5, p. e1435, 2018.

[35] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth, "Hybrid monte carlo," *Physics letters B*, vol. 195, no. 2, pp. 216–222, 1987.

[36] R. Liesenfeld and J.-F. Richard, "Improving mcmc, using efficient importance sampling," *Computational Statistics & Data Analysis*, vol. 53, no. 2, pp. 272–288, 2008.

[37] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

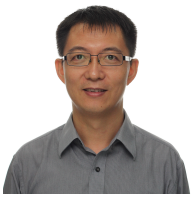
[38] W. K. Hastings, "Monte carlo sampling methods using markov chains and their applications," *Biometrika*, vol. 57, no. 1, pp. 97–109, 1970.

[39] L. Tierney, "Markov chains for exploring posterior distributions," *the Annals of Statistics*, pp. 1701–1728, 1994.

[40] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[41] A. Q. Li, A. Ahmed, S. Ravi, and A. J. Smola, "Reducing the sampling complexity of topic models," in *KDD*, 2014, pp. 891–900.

Kaiqi Zhao is a Lecturer in the School of Computer Science at the University of Auckland. He received the PhD degree from the Nanyang Technological University in 2018. Before he joined the University of Auckland, he worked as a postdoctoral research fellow in the Singtel Cognitive and Artificial Intelligence Lab for Enterprises at Nanyang Technological University. His research interests include big data analytics, geo-textual data management, and data mining.



Gao Cong is a professor in the School of Computer Science and Engineering at Nanyang Technological University (NTU). He is the director of Singtel Cognitive and Artificial Intelligence Lab for Enterprises at NTU. Prior to joining NTU, he worked with Aalborg University, Microsoft Research Asia, and the University of Edinburgh. His current research interests include geospatial and textual data management, mining social network and social media, data mining, and large scale data analytics.

Xiucheng Li is a PhD candidate in the School of Computer Science and Engineering at Nanyang Technological University. He received the MSc and BEng degrees from Harbin Institute of Technology and Harbin Engineering University, respectively. His current research interests include spatial data analytics and deep probabilistic methods.