Multithreaded and Distributed Dynamic Topic Models

Candidate Number: 28374, 31093, 38193

Abstract

Dynamic topic models (DTMs) are used to discover the time evolution of topics. To do the posterior inference, variational approximation based on Kalman filters (VKF) and a scaling up multi-thread parallelism by using Gibbs Sampling with Stochastic Gradient Langevin Dynamics (GS-SGLD) are carried out to find latent topics. In addition, the GS-SGLD algorithm is applied to a distributed platform by using Google Cloud Platform (GCP) to further speed up the computing process. By analyzing time series data of titles for COVID-19 related papers during in the last two years, the empirical results show that the parallelizable GS-SGLD not only reduces the computing time significantly in the multithreaded and distributed environment, but also achieves higher coherence score.

Contents

1 Introduction

Surrounded by a large volume of information, the demand for handling large scale information has increased. Managing such an explosion of electronic documents requires some statistical machine learning model to automatically analyse and categorise data. Some researches in 21st century has developed several useful machine learning techniques to find pattern in documents collections using some statistical model (Blei et al., 2003; McCallum et al., 2005; Rosen-Zvi et al., 2004; Griffiths and Steyvers, 2004; Buntine and Jakulin, 2004; Blei and Lafferty, 2006). Such models are called ‘Topic Modelling’, which results reflect the underlying pattern of the documents.

One of the famous models is the Latent Dirichlet Allocation (LDA), which is largely applied in the industrial and academia area. However, although LDA model give us insights about topic pattern, it requires only simplistic assumptions and gives out intractable posterior inference. It fails to realize the inference of the complicated structures under a huge volume of documents and the topic evolution in data streams (Bhadury et al., 2016).

Dynamic topic modelling (DTM) is one of the extensions to LDA that explores the evolutional trends in time series documents. The calculation of the DTM posterior inference consumes much longer time than LDA. Hence, the GS-SGLD is also a significant improvement in speeding and scaling up the algorithm.

In this report, we will implement the DTM in GCP and in PySpark Kernel, which could help us to run the application in a distributed environment. We demonstrate it by analyzing the research focus in COVID-19 related papers from the first quarter in 2020 to the first quarter in 2022. In our model, only the titles of the research papers is used, and data is sliced by quarters. The papers in the latter time slice arise from the topics that have evolved from the topics in the previous time slice.

In Section 2, previous related work is introduced. In Section 3, explicit algorithms of baseline DTM, GS-SGLD and distributed computing are explained. In Section 4, we will explain all the experiment settings as well as the experiment results. Finally in Section 5, our experiment is concluded and future improvement will be discussed.

2 Literature Review

The DTM has been particularly useful in capturing topic evolution over time due to its nature. Despite the example proposed in the original paper that tracked topics evolution of journals from Science over 100 years (Blei & Lafferty, 2006), recent works have demonstrated the application of LDA-based Dynamic Topic

Models in various research topics. For example, application of DTM in understanding the evolution of technologies (Denter et al.,2019). In social science, DTM has been used to track urban geo-topics through applying DTM on tweets over time periods (Yao & Wang, 2020). DTM has also helped to improve the forecasting accuracy of financial market volatility (Morimoto & Kawasaki, 2017).

Unlike LDA, the non-conjugacy in LDA-based DTM makes the estimation of posterior distribution challenging. The existing variational inference approach by Blei and Wang (2013) for estimating the posterior inference of DTM, which is used in Python topic modeling package Gensim, is only computationally feasible for small corpus with a small number of topics. This has limited the application of LDA-based DTM on medium to large datasets.

Bhadury et al.(2016) proposed an improved scalable inference algorithm based on Gibbs sampling with Stochastic Gradient Langevin Dynamics (SGLD) (Welling & Teh, 2011) for sampling η and Φ, and a Metropolis-Hastings based O(1) sampler for per word topic assignment Z (Bhadury et al. 2016). This algorithm makes Gibbs sampling feasible in DTM since SGLD allows convergence to the full posterior distribution without conducting Metropolis-Hastings (MH) test over the whole dataset (Welling & Teh, 2011). Because of this property, it can achieve little communication between worker nodes during training in a distributed environment. The result shows that the algorithm not only runs extremely faster than the variational inference DTM, but also has lower perplexity since SGLD provides an estimation closer to the true posterior distribution.

Gropp et al. (2019) proposed an alternative method called the Clustered Latent Dirichlet Allocation. The algorithm merges the LDA result from each individual time slice, and uses k-means clustering to produce a global solution. This method significantly reduced the running time and can be parallel to a large extent. But at the same time, it removes the time dependency structure of DTM. Since being able to predict topic evolution based on a Markov chain structure is a fundamental part of DTM, this algorithm can be viewed as a new method instead of a scaled up version of DTM.

3 Methodology

To model a time series data, Blei & Lafferty (2006) introduced DTM, which is an extension to LDA. It chains time-specific topic proportions via a Markov process underlying a logistic normal parameterization and allows evolution of parameters with a Gaussian noise. Due to the non-conjugacy of Gaussian and multinomial variables in the model, posterior inference is intractable. In the following sections, we focus on two frequently used inference approaches. One is the VKF approach, which is proposed by Blei & Lafferty (2006) and the other is the Gibbs sampling method by using GS-SGLD, which is an improved inference algorithms of DTM for capturing large dynamic topic models and presented by Bhadury et al. (2016).

3.1 Baseline DTM based on VKF

According to Blei & Lafferty (2006), we suppose the input data are divided by time slice and we model the documents in each time slice with a K component topic model. Then for each time slice t, the associated topics can be evolved from the previous topics at time slice $t-1$. The key point behind is using the mean parameterization of the topics to introduce mean chaining, so that it is possible to model time dependencies over time. Besides, to chain topics over time, DTM models the chain of mean parameters by introducing the Gaussian noise, which are used to express the uncertainty over time slices. We summarize the parameters used in DTM in Table 1.

Parameters	**Explanation**
t	time slice
d	d^{th} document at a time slice
n	n^{th} word in a document
k	k^{th} topic
T	total number of time slices
K	total number of topics
D_t	documents at time slice t
D	all input documents
$N_{d,t}$	total number of words in document d at time slice t
α_t	mean of logistic normal distribution at time slice t
$\eta_{d,t}$	topic distribution for document d at time slice t
$Z_{d,n,t}$	topic indicator, auxiliary variable
$W_{d,n,t}$	observation value, n^{th} word in the document d at time slice t
$\Phi_{k,t}$	distribution over words for topic k at time slice t
Φ_t	topics at time slice t
σ, β, ψ	variance parameters in Gaussian distributions
π	map function, from multinomial parameters to the mean parameters

Table 1: Parameters used in DTM .

Given a set of documents $D = (D_t)_{t=1}^{T}$, for each time slice t, DTM first draws evolved parameters mean α_t of the logistic normal distribution, which is used to express the uncertainty over document-specific topic properties. And sampling topic-term proportion $\Phi_{k,t}$ for each topic k at time t from Gaussian noises.

For each time slice t:

1. Draw evolved parameter mean $\alpha_t|\alpha_{t-1} \sim \mathcal{N}(\alpha_{t-1}, \sigma^2 I)$
2. Draw words distribution $\Phi_{k,t}|\Phi_{k,t-1} \sim \mathcal{N}(\Phi_{k,t-1}, \beta^2 I)$, $\forall k \in [K]$

Then for each document d at time slice t, topic proportion $\eta_{d,t}$ is chained by using the evolved parameter α_t. Next, for n^{th} word in document d, topic indicator $Z_{d,n,t}$ is drawn from multinomial distribution by using the mean parameter $\pi(\eta_{d,t})$, and then the word $W_{d,n,t}$ can be drawn from multinomial distribution by using the mean parameter $\pi(\Phi_{Z_{d,n,t},t})$. The generative process for words is shown in Algorithm 1, and note that π maps the multinomial nature parameters $\overrightarrow{\phi}$ to the mean parameters $\overrightarrow{\gamma}$ (i.e. $\pi(\overrightarrow{\phi}) = \overrightarrow{\gamma}$) by using the formula below:

$$\gamma_i = \frac{\exp(\phi_i)}{\sum_j \exp \phi_i} \tag{1}$$

Algorithm 1 process of generating words in DTM

```
for each time slice $t$ do
    for each document $d$ do
        draw topic distribution $\eta_{d,t} \sim \mathcal{N}(\alpha_t, \psi^2 I)$
        for each word at position $n$ do
            sample topic indicator $Z_{n,d,t} \sim Multi(\pi(\eta_{d,t}))$
            sample word $_{d,n,t} \sim Multi(\pi(\Phi_{Z_{d,n,t}t}))$
        end for
    end for
end for
```

From the graphical model in Figure 1, it is obviously to see that the k^{th} topic as slice t is evolved from previous k^{th} at time slice $t-1$ under time dynamic. However, if we broke the time dynamics by removing the horizontal arrows and just focus model at time slice t, we found that the DTM reduces an independent topic model, whose structure is exactly same as LDA topic model.

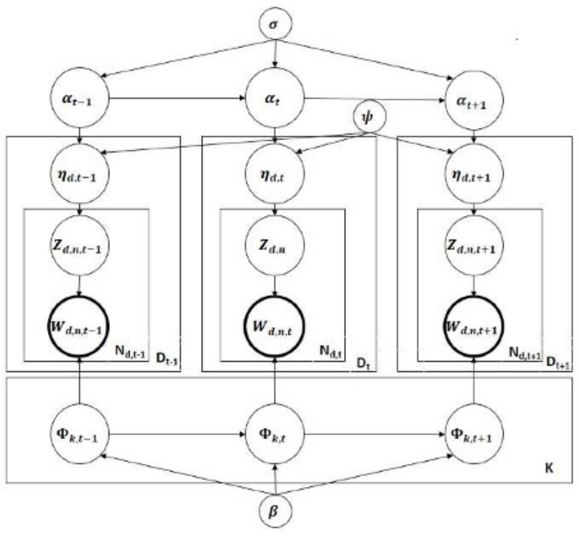


Figure 1: Graphical model for generative process of DTM (Bhadury et al.,2016)

After observing all the documents D, the conditional posterior distribution is given in the below equation. But due to the non-conjugacy of the Gaussian multinomial variables, topic-term proportion $\Phi_{k,t}$ and topic proportion $\eta_{d,t}$ cannot be integrated out. Blei & Lafferty (2006) explored two variational methods, Kalman filtering, which considers the symmetrical properties of the Gaussian density to yield variational parameters, and variational wavelet regression. In the following paragraphs, we are going to explain VKF method in detail.

$$\begin{aligned} p(\alpha, \eta, \Phi, Z | D, \theta, \beta, \psi) \\ &\propto \prod_{t=1}^{T} \mathcal{N}(\alpha_{t-1}, \sigma^2 I) \times \prod_{k=1}^{K} \mathcal{N}(\Phi_{k,t-1}, \beta^2 I) \times \prod_{d=1}^{D_t} \eta_{d,t} \sim \mathcal{N}(\alpha_t, \psi^2 I) \\ &\times \prod_{n=1}^{N_{d,t}} Multi(\pi(\eta_{d,t})) Multi(\pi(\Phi_{Z_{d,n,t}t})) \end{aligned} \tag{2}$$

To retain the sequential structure of topics over time, DTM fits a dynamic model with Gaussian variational observations $(\hat{\Phi}_{k,1}, ..., \hat{\Phi}_{k,t}, ...\hat{\Phi}_{k,T})$, fitting these parameters to minimize the Kullback-Leibler divergence between the resulting posterior and the true posterior. Then DTM uses Kalman filtering, which enables the use of backward-forward calculations in a linear state space model, to mimic Gaussian variational observations.

Forward-backward calculations on Kalman parameters allows to estimate posterior mean m_t and variance parameter V_t in terms of Gaussian parameters over topics. In detailed, as time sliced topics Φ_t are conditioned on the previous sliced topic Φ_{t-1}, and both are related by a Gaussian of β parameter, the variational state space model $\hat{\Phi}_t$ is conditional on $\hat{\Phi}_{t-1}$ and both are related by a Gaussian of $\hat{v}_t$ parameter. Then in forward calculations, posterior mean m_t and variance parameter V_t are calculated from β and from the variational parameters $\hat{\Phi}_t$ and $\hat{v}_t$ (Kalman, 1960). In the backward calculations, the marginal mean $\widetilde{m}_{t-1}$ and variance $\widetilde{V}_{t-1}$ depends on posterior mean m_{t-1} and variance V_{t-1}, β and the one-step ahead marginal mean $\widetilde{m}_t$ and variance $\widetilde{V}_t$. Forward calculations use initial conditions m_0 and V_0 while backward calculations use initial conditions $\widetilde{m}_T = m_T$ and $\widetilde{V}_T = V_T$. The detailed formulas can be found in Appendix. the rest of the parameters are estimated using variational expectation maximization as was used in the original posterior inference of LDA (Blei et al., 2003)

3.2 Scaling up DTM based on GS-SGLD algorithm

3.2.1 SGLD approach for posterior estimation

Let θ be a parameter vector with prior $p(\theta)$ and posterior $p(x|\theta)$ of a data set $\boldsymbol{X} = x_1, ..., x_N$. The SGLD takes into account the advantage of Langevin dynamics algorithms in capturing uncertainty. The injected Gaussian noise ensures more samples are explored by the algorithm instead of converging to the MAP (Patterson & Teh, 2013). In Langevin dynamics, the gradient step sizes balanced the variances of the injected Gaussian noise, so the variance of the samples matches the variance of the posterior (Welling & Teh, 2011).

SGLD uses the mini-batch technique of Stochastic Gradient Decent (SGD), instead of calculating the log-likelihood w.r.t θ using the whole dataset as in Langevin dynamics. The parameter update at each iteration t is as following:

$$\Delta\theta_t = \frac{\epsilon_t}{2}\Big(\nabla \log p(\theta_t) + \frac{N}{D_t}\sum_{n=1}^{D_t} \nabla \log p(x_{ti}|\theta_t)\Big) + \xi_t \tag{3}$$

where $\xi_t \sim N(0, \epsilon_t)$ is the Gaussian noise assigned during each iteration and x_{ti} comes from the mini-batch of size D_t at each iteration t.

The step size is defined as $\epsilon_t = a(b+t)^{-c}$, with $c \in (0.5, 1]$ to satisfy the following two necessary conditions for convergence to local maximum in SGD: $\sum_{t=1}^{\infty} \epsilon_t = \infty$ and $\sum_{t=1}^{\infty} \epsilon_t^2 < \infty$. Welling and Teh (2011) have proved that under this condition, for large enough t,$\epsilon_t \to 0$ and the injected noise $\xi_t \sim N(0, \epsilon_t)$ will dominate the stochastic gradient noise with variance $(\frac{\epsilon_t}{2})^2 V(\theta_t)$, making the whole algorithm Langevin dynamics, thus converges to the posterior distribution. Welling and Teh also show that in the context of $\epsilon_t \to 0$, MH rejection probability becomes close to 0 so there is no need to correct discretization error using the whole data set. This improvement, alongside with the mini-batch technique of SGD, makes the algorithm a lot more faster and scalable, leaving possibilities for parallel and distributed computation. The Langevin dynamics nature of SGLD allow it to explore more samples from the posterior distribution, which outperforms VKF, who's data size and topic number are limited due to its inability to explore large amount of samples.

3.2.2 Gibbs sampling in DTM

SGLD approach mentioned before is employed in Gibbs Sampling of unknown parameters $\eta_{d,t}$ and $\Phi_{k,t}$, treating these parameters as the parameter θ in 3.

The per word topic assignment $Z_{d,n,t}$ is conditionally independent of the rest of the parameters, it's posterior distribution can be written as the product of $\exp(\eta_{d,t})$ and $\exp(\Phi_{k,t})$, corresponding to the generators of document and word proposals. An algorithm bases on LightLDA (Yuan et al.,2014) is used to bring the MH test complexity down to $O(1)$, while still maintain a high acceptance rate. This is achieved by realising that if the posterior has high probability mass on topic k, then either the word proposal or document proposal will also have high probability mass on k, so both of them serve as good proposals for MH test (Yuan et al.,2014). By transforming the sampling of both the word proposal and document proposal to a Uniform distribution sampling using the Alias table (Marsaglia, Tsang & Wang, 2004), the MH complexity is then reduced to $O(1)$.

3.3 Multithreaded and Distributed Computation

3.3.1 Distributed Computing

Distributed computing refers to a computer system consists of multiple computers as workers running a single whole program (IBM, 2021). To realize the cooperative work of multiple computers, the computers in the distributed system are divided into two roles, master and workers. The master receives the user program and creates a set of tasks to initiates the computation, and then assign the task to each worker as Figure 2. At the very end of the task, the master also collects the results from each worker and terminate the computation. Such kind of pattern could automatically balance the load for each worker. On the other hand, it usually has good scalability when the tasks take similar time to complete.

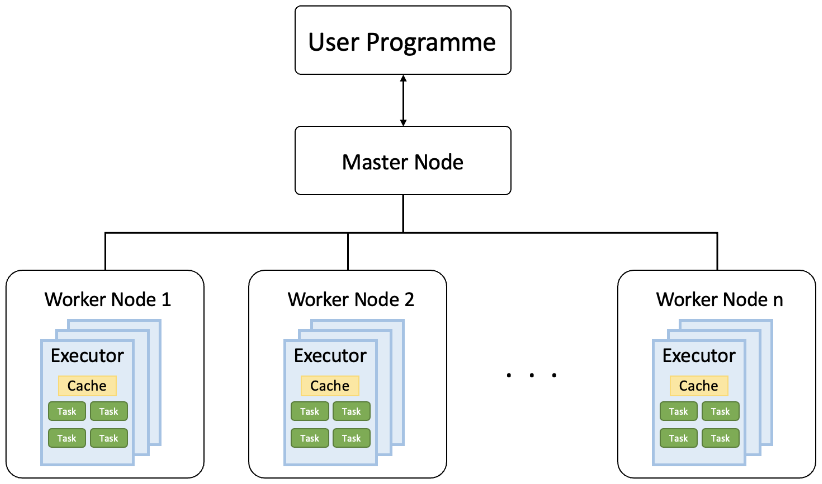
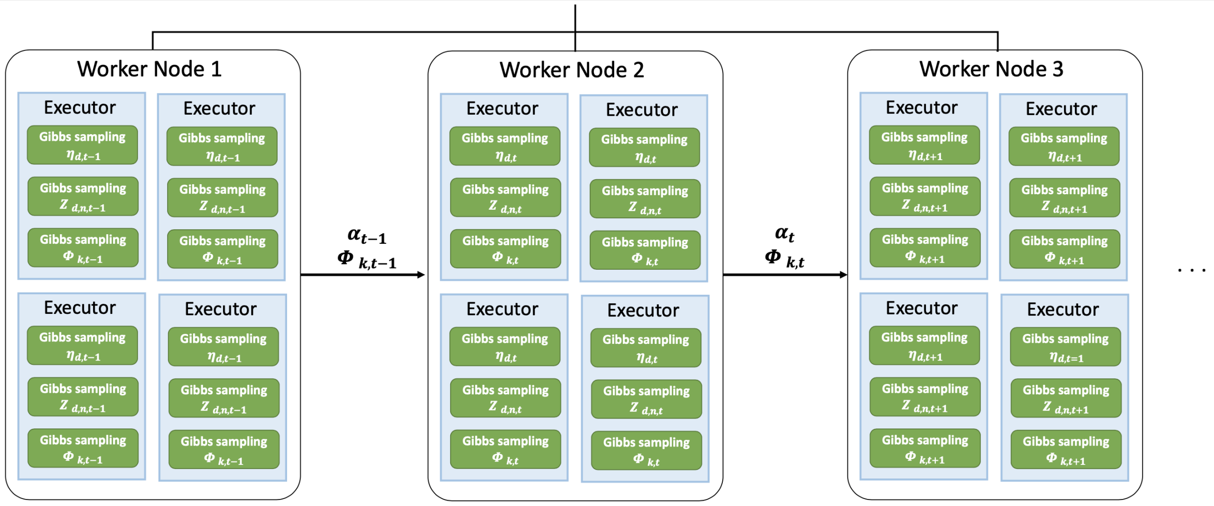


Figure 2: Distributed Structure

Our whole algorithm is applied on GCP, which is a public cloud provider, providing a distributed environment. Both Hadoop and Spark are embedded in GCP. Hadoop is a popular implementation for MapReduce and designed for executing workflows on clusters of machines (Galakatos et al., 2018). It could provide a distributed file system that allows users analyzing large scales of data across hundreds of machines. Spark is also a distributed framework (Galakatos et al., 2018). It allows users to implement workflows using predefined API operators in Python and could sufficiently support iterative algorithms, which is quite suitable for GS-SGLD. In this environment, the algorithm could be distributed to several worker nodes. Ideally, the Markorvian nature in the model will pass α_t and Φ_t to adjacent nodes (Bhadury et al.2016) until some criteria has been reached to end the computation. Such kind of embarrassingly parallelism could effectively improve the computing time performance.

3.3.2 Multithreaded Computing

Within each worker node, parallel computing is also applied. The detailed workflow is illustrated by Figure 3. The pseudo-code for the Approximate Parallel Algorithm (Newman et al., 2009) is shown in Algorithm 2. After distributing the data and parameters across workers, the parallel algorithm helps to perform Gibbs sampling simultaneously. Then each executor within the worker node has swept through its local data and updated the model topic assignment z_p. The word-topic counts $N_{d,t,p}$ is generally different on different executors and not consistent in the worker. Then the reduce operation is applied to merge results back to single. The whole algorithm could be terminated after a predefined number of maximum iterations or met specific predefined criteria such as the convergence metric.



Worker Node 1
Worker Node 2
Worker Node 3
Executor
Gibbs sampling $\eta_{d,t-1}$
Gibbs sampling $Z_{d,n,t-1}$
Gibbs sampling $\Phi_{k,t-1}$
Gibbs sampling $\eta_{d,t}$
Gibbs sampling $Z_{d,n,t}$
Gibbs sampling $\Phi_{k,t}$
Gibbs sampling $\eta_{d,t+1}$
Gibbs sampling $Z_{d,n,t+1}$
Gibbs sampling $\Phi_{k,t+1}$
α_{t-1} $\Phi_{k,t-1}$
α_t $\Phi_{k,t}$
...

Figure 3: parallelism inside the working node

Algorithm 2 Approximate Parallel

```
for each processor $p$ in parallel do
    Copy global counts: $N_{d,t,p} \leftarrow N_{d,t}$
    Sample $\eta_p, z_p, \Phi_p$ locally using Gibbs iteration
end for
Synchronize
Update global counts $N_{d,t} \leftarrow N_{d,t} + \sum(N_{d,t,p} - N_{d,t})$
```

On the other hand, it is only an approximation since in this algorithm, we are not sampling from the true posterior but an approximation of the true posterior. Nonetheless, Newman et al. (2009) have showed that the approximation worked by this algorithm performs very well.

4 Experiments

4.1 Datasets and Setups

We analyzed a subset of 429581 titles of coronavirus-related research papers from January 2020 to March 2021. Our data was collected by the Semantic Scholar team (Wang et al., 2020), and it is an COVID-19 open research dataset for mobilizing researchers to apply recent advances in natural language processing to generate new insights to support this infectious disease. Based on the dynamic topic model, we can observe how the thematic structure of coronavirus research evolved over time.

To quantitatively compare the baseline dynamic topics model (Blei & Lafferty, 2006) based on a VKF approach to a multithreaded dynamic topic model (Bhadury et al. 2016) based on GS-SGLD, we present our models on a single machine in a multithreaded environment in section 4.2. We use n1-standard-4 (i.e. 4-vCPU, 15GB memory) node without any workers based on the Dataproc cluster on GCP to do the first experiment. The dataset used here consists of all titles of CORD-19 research papers during the first quarter in 2020, where we have 8074 titles as input documents and 11337 words as the vocabulary with stop words removed. And we divided this dataset into three time slices by month.

On the other hand, we also present our parallel GS-SGLD model, which is scalable to a large dataset, in the distributed environment and the results are shown in section 4.3 and 4.4. To quantitate the performance of multi-machines distributed computation version and multi-threads parallelism, we consider three environments here: single machine with 2 YARN cores, single machine with 4 YARN cores and 4-workers cluster consisting of 16 YARN cores in total. In detailed, we still run the tasks on Dataproc cluster, correspondingly using the n1-standard-2 and n1-standard-4 node with 0 worker as the first two environments and using the 4-vCPU master node with 4 n1-standard-4 workers as our distributed computing platform. Also, we use a large dataset here to compare the computing performance, which consists of all titles of coronavirus related papers from April 2020 to March 2021, during which covariance researches growth spurt. This dataset contains more than 420 thousand documents and our corpus is made up of 3526110 words in total. We pruned the vocabulary by stemming each term to its root, removing numbers and punctuation, removing single letters, and removing stop words along with some useless words. The total number of vocabulary size is 130651. To explore the themes during this COVID period, we estimated an 8 time slices dynamic topic model, where corpus are divided by quarters.

4.2 Comparison of GS-SGLD and VKF method

In this section, we compare GS-SGLD model to the baseline DTM of using VKF method. We run GS-SGLD algorithm by using DTM model in ‘tomotopy’ package, which is based on Bhadury et al. (2016), on the small dataset containing a total of 8074 titles divided into 3 time slices by month. In sampling process, the hyper-parameter we used are shown in Table 2. Besides, we use Copy and Merge algorithm to parallel sample respectively for each time slice. For baseline VFK method, we use LdaSeq model in ‘genesim’ package, which is according to Blei and Lafferty (2006), and the parameters were set as in Table 3. We use the same documents and capture 5 themes for each month.

Parameters	**Explanation**	**Value**
alpha_var	transition variance of α (mean of the logistic normal distribution)	0.1
eta_var	transition variance of η (topic distribution of each document) from its α	0.1
phi_var	transition variance of Φ (word distribution of each topic)	0.01
lr_a	a, b are parameters used to decay ϵ from 0.01 to 0.0001 over the run SGLD step size $\epsilon_t = a \times (b+t)^{-c}, c \in (0.5, 1]$	0.01
lr_b		0.1
lr_c		0.55
seed	random seed	0
ParallelScheme	parallel algorithm used in Gibbs Sampling	COPY_MERGE

Table 2: Used parameters for DTM model in 'tomotopy' package.

Parameters	**Explanation**	**Value**
alphas	the prior probability for the model	0.01
initialize	initialization of the DTM model	'gensim'
lda_inference_max_iter	maximum number of iterations in the inference of the LDA training	100
em_min_iter	minimum number of iterations until coverage of the EM algorithm	6
em_max_iter	maximum number of iterations until converge of the EM algorithm	20
random_state	random seed	0

Table 3: Used parameters for LdaSeq model in 'genesim' package.

According to Zvornicanin (2021), we can use the coherence score in topic modeling to measure how interpretable the topics are to humans. In this case, topics are represented as the top words with the highest probability of belonging to that particular topic. Briefly, the coherence score measures how similar these words are to each other. Here we use UMass coherence to compare the performances of different models. It calculates how often two words appear together in the corpus and it's defined as following formula:

$$C_{UMass}(w_i, w_j) = log\tfrac{D(w_i,w_j)+1}{D(w_i)} \tag{4}$$

where $D(w_i, w_j)$ indicates how many times word w_i and w_j appear together in documents and w_i indicates how many times word w_i appears alone.

We calculate the global coherence of the topic as the average pairwise coherence scores on the top 20 words which describe the topic. Besides, the greater the number, the better the coherence score.

	Month	**VKF**	**GS-SGLD**
Coherence Score (UMass)	**1**	-6.20	-4.97
	2	-6.15	-4.59
	3	-6.34	-4.60
Computation Time (s)		2087.64	36.02
CPU utilization (single machine with 4-vCPU)		27.5%	64.0%

Table 4: Results comparison of GS-SGLD and VKF method.

Table 4 shows results of the GS-SGLD algorithm compared to the baseline DTM when capturing 5 topics for the small dataset. We find that GS-SGLD is significantly faster than the VKF method because of using mini-batch during gradient steps and the amortized sampler for the topics of each token. In addition, we observe that GS-SGLD achieves a higher UMass coherence for each time slice, compared to VKF. It shows that the performance of GS-SGLD seems much better than baseline DTM, at least from the coherence score aspect. Furthermore, the CPU usage in GS-SGLD is much higher than in VKF method due to the efficiency of parallel sampler during the Gibbs sampling process by optimizing the CPU cache access.

Therefore, Gibbs Sampler is not only faster than VKF approach, but also has better topic performance based on UMass coherence score. Furthermore, according to Bhadury et al (2016), GS-SGLD is efficient in capturing large dynamic topic models, so we apply the GS-SGLD algorithm to the large dataset in the following sections.

4.3 Topic Evolution of DTM

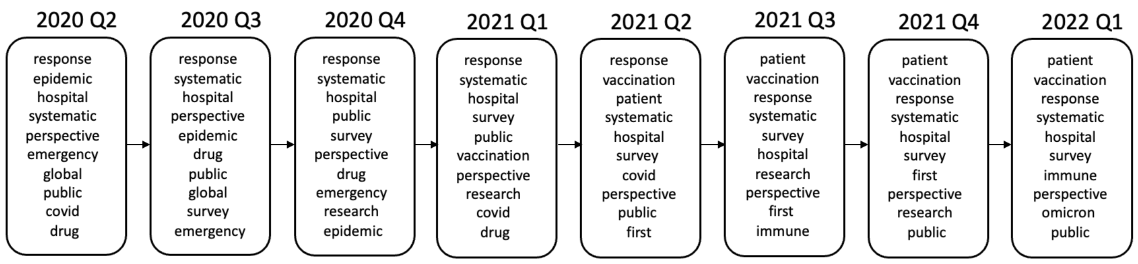

2020 Q2
response epidemic hospital systematic perspective emergency global public covid drug
2020 Q3
response systematic hospital perspective epidemic drug public global survey emergency
2020 Q4
response systematic hospital public survey perspective drug emergency research epidemic
2021 Q1
response systematic hospital survey public vaccination perspective research covid drug
2021 Q2
response vaccination patient systematic hospital survey covid perspective public first
2021 Q3
patient vaccination response systematic survey hospital research perspective first immune
2021 Q4
patient vaccination response systematic hospital survey first perspective research public
2022 Q1
patient vaccination response systematic hospital survey immune perspective omicron public

Figure 4: Topic Evolution of DTM

Figure 4 above shows one of the five sets of topic evolution results over eight quarters, based on the posterior estimate using GS-SGLD. Each block shows the top 10 weighted topic words of that quarter. The model captures the trend of word usage within each set of topics. For this topic, the ranking of top weighted words changes in each quarter. “response” has been the top weighted word within the topic from 2020 Q2 to 2021 Q2, and it has been replaced by “patient” since 2021 Q3. It is also observed that the word “epidemic” has been frequently used at the start, but vanished from top 10 since 2021 Q1. The word “vaccination” appeared since 2021 Q1 and has been frequently discussed in later time slices. The word “omicron” appeared in the top 10 in 2022 Q1, while the Omicron variant was first detected in early November 2021.

4.4 Multithreaded and Distributed Comparison

In this section, all the model parameters are the same as them in Table 1. In order to compare the computational efficiency of 2-core single machine, 4-core single machine and distributed setting with 4 workers, the model is trained under different settings and different number of time slices for 5 times. Then the average time is computed.

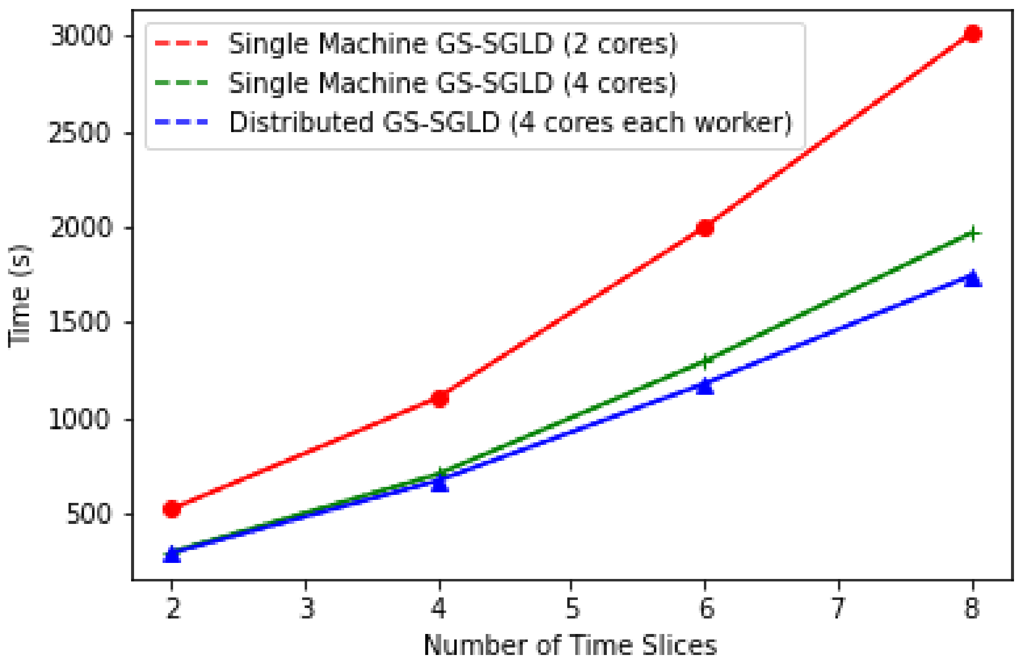


Figure 5: Time Comparison among different settings.

Figure 5 illustrates the time performance against different numbers of time slices among three settings. The red line is the performance of the single machine with 2 cores, the green line represents the single machine with 4 cores, and the performance of the distributed setting is illustrated by the blue line. From the plot, the distributed GS-SGLD has the best time performance and the single machine with only two cores always consumes the longest time in all numbers of time slices.

Compared with 2-core single machine, 4-core single machine trained the model much faster since there is a significant gap between red and green line. As the number of time slices goes up, the 4-core single machine saves more time. When all the data is used to train the model, a 4-core single machine is about 17 minutes (1000 seconds) faster than the 2-core single machine.

When it comes to the distributed setting, the training performance in training time gets another boost. In each time slice, the blue line is below the green line. Similarly, the more time slices there is, the more time is saved by the distributed setting.

Both comparisons indicate that the algorithm scales up well with the increase of time slices due to its parallel nature.

From the comparison above, both parallelism and distributed computing could improve computing performance. However, distributed computing does not improve as much as parallel computing. With only 2 time slices, the distributed computing is only 4 seconds faster than the 4-core single machine while the 4-core single machine is 225 seconds faster than 2-core single machine. Despite the training time, the average CPU utilization is also recorded in the Table 4.

Average CPU utilization in training process	Single Machine 2 YARN cores (master: 2-vCPU, 7.5GB memory, 0 workers)	Single Machine 4 YARN cores (master: 4-vCPU, 15GB memory, 0 workers)	Distributed Machine 16 YARN cores (master: 4-vCPU, 15GB memory, 4 workers: 4-vCPU, 15GB memory)
2 Time Slice	96.5%	91.0%	18.7%
4 Time Slice	95.3%	89.0%	18.3%
6 Time Slice	95.0%	87.3%	17.9%
8 Time Slice	94.6%	85.5%	17.3%

Table 5: Average CPU utilization

From Table 5, we notice that the CPU utilization significantly decreased in the distributed setting. The

CPU utilization for both single machine cluster settings are above 85% while the distributed only use less than 20% usage of the CPU. The low CPU usage could be the reason for the unsatisfactory improvement of the distributed setting, which means if there are any methods that could increase the CPU usage, the computing time of the distributed setting will be much lower than it is now.

5 Discussion and future work

We compared the running time and topic coherence of the variational inference-based DTM and the Gibbs-sampling based one on a single machine, using a small subset of the dataset. We then demonstrated the difference in running time when applying the Gibbs-sampling based DTM on a single machine vs a distributed environment using all data. Our result shows that the Gibbs-sampling based DTM requires much less running time and has higher topic coherence. Also, the running time can be further reduced when the model is applied in a multithreaded and distributed environment. Our two results align with those obtained by Bhadury et al.(2016).

There are two limitations associate with our research. Firstly, when we applied GS-SGLD on distributed machine with four workers, the average CPU utilization is poor. This might be due to the default settings in GCP. If we can find a way to utilize more CPU usage, the reduction in running time of parallel and distributed DTM might be more prominent. Secondly, we used the paper titles instead of all the words in the documents(papers) due to limited computing power. This results in a smaller vocabulary set. If we have used all the words in the documents, the topics generated will be more accurate, and we will obtain much better results. In future research, we can implement the algorithm by using all the words of the papers and compare the results.

6 Appendix

Formulas of forward mean and variance of the variational posterior in VKF:

$$m_t \equiv \mathbb{E}(\Phi_t|\hat{\Phi}_{1:t}) = (\frac{\hat{v_t}^2}{V_{t-1} + \beta^2 + \hat{v_t}^2})m_{t-1} + (1 - \frac{\hat{v_t}^2}{V_{t-1} + \beta^2 + \hat{v_t}^2})\hat{\Phi}_t \tag{5}$$

$$V_t \equiv \mathbb{E}((\Phi_t - m_t)^2|\hat{\Phi}_{1:t}) = (\frac{\hat{v_t}^2}{V_{t-1} + \beta^2 + \hat{v_t}^2})(V_{t-1} + \beta^2) \tag{6}$$

Formulas of marginal mean and variance in backward recursion for VKF:

$$\widetilde{m}_{t-1} \equiv \mathbb{E}(\Phi_t|\hat{\Phi}_{1:t}) = (\frac{\beta^2}{V_{t-1} + \beta^2})m_{t-1} + (1 - \frac{\beta^2}{V_{t-1} + \beta^2})\widetilde{m}_t \tag{7}$$

$$\widetilde{V}_{t-1} \equiv \mathbb{E}((\Phi_t - \widetilde{m}_{t-1})^2|\hat{\Phi}_{1:t}) = V_{t-1} + (\frac{V_{t-1}}{V_{t-1} + \beta^2})(\widetilde{V}_t - (\widetilde{V}_{t-1} + \beta^2)) \tag{8}$$

References

[1] Blei, D.M. and Lafferty, J.D., 2006. Dynamic Topic Models Proceedings of the 23rd International Conference on Machine Learning (ICML'06). 113–120. *Google Scholar Google Scholar Digital Library Digital Library.*

[2] Blei, D.M., Ng, A.Y. and Jordan, M.I., 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan), pp.993-1022.

[3] Bhadury, A., Chen, J., Zhu, J. and Liu, S., 2016, April. Scaling up dynamic topic models. *In Proceedings of the 25th International Conference on World Wide Web*, pp. 381-390.

[4] Buntine, W. and Jakulin, A., 2004. Applying discrete PCA in data analysis. *In Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, pages 59–66. AUAI Press.

[5] Denter, N., Caferoglu, H. and Moehrle, M.G., 2019, August. Applying Dynamic Topic Modeling for Understanding the Evolution of the RFID Technology. *In 2019 Portland International Conference on Management of Engineering and Technology (PICMET)*, pp. 1-9. IEEE.

[6] Galakatos, A., Crotty, A. and Kraska, T., 2018. Distributed Machine Learning.

[7] Griffiths, T. and Steyvers, M., 2004. Finding scientific topics. *Proceedings of the National Academy of Science*, 101:5228–5235.

[8] Gropp, C., Herzog, A., Safro, I., Wilson, P.W. and Apon, A.W., 2019, December. Clustered latent dirichlet allocation for scientific discovery. *In 2019 IEEE International Conference on Big Data*, pp. 4503-4511. IEEE.

[9] Kalman, R.E., 1960. A new approach to linear filtering and prediction problems.

[10] Marsaglia, G., Tsang, W.W. and Wang, J., 2004. Fast generation of discrete random variables. *Journal of Statistical Software*, 11, pp.1-11.

[11] McCallum, A., Corrada-Emmanuel, A. and Wang, X., 2005. The author-recipient-topic model for topic and role discovery in social networks: Experiments with enron and academic email. *Computer Science Department Faculty Publication Series*, p.44.

[12] Hoffman, M.D., Blei, D.M., Wang, C. and Paisley, J., 2013. Stochastic variational inference. *Journal of Machine Learning Research.*

[13] Morimoto, T. and Kawasaki, Y., 2017. Forecasting financial market volatility using a dynamic topic model. *Asia-Pacific financial markets*, 24(3), pp.149-167.

[14] Newman, D., Asuncion, A., Smyth, P. and Welling, M., 2009. Distributed algorithms for topic models. *Journal of Machine Learning Research*, 10(8).

[15] Patterson, S. and Teh, Y.W., 2013. Stochastic gradient Riemannian Langevin dynamics on the probability simplex. *Advances in neural information processing systems*, 26.

[16] Rosen-Zvi, M., Griffiths, T., Steyvers, M., and Smith, P., 2004. The author-topic model for authors and documents. *In Proceedings of the 20th Conference on Un- certainty in Artificial Intelligence*, pages 487–494. AUAI Press.

[17] Wang, L.L., Lo, K., Chandrasekhar, Y., Reas, R., Yang, J., Eide, D., Funk, K., Kinney, R., Liu, Z., Merrill, W. and Mooney, P., 2020. Cord-19: The covid-19 open research dataset. ArXiv.

[18] Welling, M. and Teh, Y.W., 2011. Bayesian learning via stochastic gradient Langevin dynamics. *In Proceedings of the 28th international conference on machine learning* (ICML-11), pp. 681-688.

[19] www.ibm.com. (2015). What is distributed computing. Available at: https://www.ibm.com/docs/en/txseries/8.2?topic=overview-what-is-distributed-computing.

[20] Yao, F. and Wang, Y., 2020. Tracking urban geo-topics based on dynamic topic model. *Computers, Environment and Urban Systems*, 79, p.101419.

[21] Yuan, J., Gao, F., Ho, Q., Dai, W., Wei, J., Zheng, X., Xing, E.P., Liu, T.Y. and Ma, W.Y., 2015, May. Lightlda: Big topic models on modest computer clusters. *In Proceedings of the 24th International Conference on World Wide Web*, pp. 1351-1361.