Fast Topic Discovery From Web Search Streams

Di Jiang, Kenneth Wai-Ting Leung, Wilfred Ng
Department of Computer Science and Engineering
Hong Kong University of Science and Technology, Hong Kong, China
{dijiang, kwtleung, wilfred}@cse.ust.hk

ABSTRACT

Web search involves voluminous data streams that record millions of users' interactions with the search engine. Recently latent topics in web search data have been found to be critical for a wide range of search engine applications such as search personalization and search history warehousing. However, the existing methods usually discover latent topics from web search data in an offline and retrospective fashion. Hence, they are increasingly ineffective in the face of the ever-increasing web search data that accumulate in the format of online streams. In this paper, we propose a novel probabilistic topic model, the *Web Search Stream Model* (WSSM), which is delicately calibrated for handling two salient features of the web search data: it is in the format of streams and in massive volume. We further propose an efficient parameter inference method, the *Stream Parameter Inference* (SPI) to efficiently train WSSM with massive web search streams. Based on a large-scale search engine query log, we conduct extensive experiments to verify the effectiveness and efficiency of WSSM and SPI. We observe that WSSM together with SPI discovers latent topics from web search streams faster than the state-of-the-art methods while retaining a comparable topic modeling accuracy.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Search process

General Terms

Design, Experimentation, Performance

Keywords

Web Search; Query Log; Probabilistic Topic Model

1. INTRODUCTION

Web search usage data [3] is the embodiment of millions of search engine users' underlying information needs. Researchers have found that latent topics in web search data are effective for improving the performance of a wide spectrum of search engine applications such as search personalization [6, 29], location-based services [15], search history warehousing [18] and entity mining [23]. Several probabilistic topic models have been developed to discover latent topics from web search data in an offline and retrospective fashion [6, 15, 18, 23, 29]. However, as web search data is essentially in the format of massive streams [10, 25], the existing models are becoming ineffective due to their offline nature as well as their limited capability of processing voluminous data.

Table 1: Web Search Stream of A Search Engine User

ID	Search Query	Clicked URLs
1	hotel chicago	www.expedia.com/...
2	chicken run	
3	chicken run movie	www.imdb.com/...
4	fuji mountain tour	www.japan-guide.com/...
5	fuji travel	www.japanican.com/...
...	...	...

From the perspective of naturally modeling web search data, a search engine user's web search activities should be viewed as an online stream of search sessions, which is referred to as a series of consecutively submitted search queries and clicked URLs that satisfy a single information need of a specific search engine user [16]. For example, in Table 1, the user's web search stream contains infinite entries that are chronologically ordered and each entry contains a search query and its corresponding clicked URLs (if any). The first entry itself forms a search session, the second and third queries are of a search session while the fourth and the fifth entries belongs to another search session. Besides viewing web search data from a more natural perspective, the efficiency of training the model is another major concern when analyzing web search data. For real-life applications, short response-time is usually critical [7] and latent topics need to be discovered from massive web search streams by consuming fairly short time. Thus, it is necessary to investigate how to develop an effective topic model and an efficient parameter inference method, which collectively are able to digest millions of web search streams in short time span while still maintaining high topic modeling accuracy. The challenges are essentially twofold:

1. Web search streams contain heterogeneous items (i.e., query words and clicked URLs), hierarchical structures (i.e., search sessions) and are dynamical evolving. How to design a model that is able to effectively handle the above features is still an open problem.

2. In the face of the voluminous size of web search data, training efficiency is critical for real-life applications. How to

WWW'14, April 7–11, 2014, Seoul, Korea.
ACM 978-1-4503-2744-2/14/04.
http://dx.doi.org/10.1145/2566486.2567965 .

design an efficient and accurate parameter inference method that is workable on web search streams is rarely explored in the literature.

To address the aforementioned challenges, we propose the *Web Search Stream Model* (WSSM), a probabilistic topic model delicately calibrated for discovering latent topics from massive web search streams with high accuracy. WSSM captures the information coherency within each search session and models the ternary relations between search sessions, query words and clicked URLs in a principled way. Besides discovering latent topics from web search streams, WSSM is able to detect topic evolution over time. Confronted with the massive size of web search streams, conventional parameter inference methods such as collapsed Gibbs Sampling (GS) [24] and Variational Bayes (VB) [2] are inefficient for training WSSM because of their high computational cost. As GS and VB typically require multiple iterations of scanning the entire corpus of data and the complete topic space, their computational cost usually increases linearly with the data size, the number of topics and the number of training iterations. In order to solve this problem, we propose a fast parameter inference method named *Stream Parameter Inference* (SPI). SPI utilizes the *Web Search Matrix* (WSM) for reducing the amount of data need to be processed by downstream operation. SPI further combines belief propagation [20] with the stochastic gradient descent framework [4], which ensures that the parameter inference process converges to the stationary point of the likelihood function of WSSM by a series of online gradient updates. In each iteration, SPI actively selects a fraction of web search data and a part of the topic space for message updating and passing, and this paradigm significantly speedups the parameter inference process. Based on a real-life query log, we conduct a series of evaluations to verify the effectiveness of WSSM and the efficiency of SPI. We observe that WSSM demonstrates superior capability of discovering latent topics from web search streams. Compared with several existing parameter inference methods, SPI is more efficient for training WSSM in term of both time and memory consumption.

The major contributions of this paper are summarized as follows:

- We propose a new probabilistic topic model, the *Web Search Stream Model* (WSSM), to effectively discover latent topics from massive web search streams. WSSM handles two salient features of web search data: it is in the format of streams and in massive volume.
- We develop a novel parameter inference algorithm, the *Stream Parameter Inference* (SPI), to efficiently train WSSM. SPI is built upon belief propagation and selects a fraction of web search data and a part of the topic space to train WSSM with both high efficiency and high accuracy.
- We conduct extensive experiments to compare WSSM and SPI with several the state-of-the-art probabilistic topic models and parameter inference methods. The experimental results show that WSSM and SPI outperform their corresponding baselines with respect to a variety of metrics.

The rest of the paper is organized as follows. We review the most related work in Section 2. In Section 3, we discuss the assumptions and generative process of the *Web Search Stream Model* (WSSM). In Section 4, we discuss the technical details of *Stream Parameter Inference* (SPI) such as how to utilize SPI to train web search data within a time period as well as that within a span of time periods. We present the experimental results in Section 5 and conclude the paper in Section 6.

2. RELATED WORK

The present work is closely related to a wide range of existing works that apply probabilistic models to web search and microblog data. Probabilistic models have achieved promising performance in different search engine applications. [27] proposed a conditional random fields model for user intent learning from search sessions. [5] proposed a hidden Markov model to facilitate context-aware search. [33] described an application of partially observable Markov model to analyze a large-scale query log. Among the diverse types of probabilistic models, topic models are found to be an effective tool for query log analysis [6, 15, 17, 23]. For example, [17] proposed a topic-concept cube that supports online multidimensional mining of query log. [34] presented a topic model that captures latent structure of textual data and how the structure changes over time. More recently, with the popularity of microblogs, researchers are aware of the importance of analyzing text streams in real-life applications. [19] presented a topic model to track emerging events in microblog data such as tweets. [13] presented an algorithm to model diversity phenomena in tweet streams based on topical diversity and geographical diversity. [9, 8] described how to efficiently capture the statistics of stream data. However, to the best of our knowledge, none of the existing techniques are primarily proposed for processing massive web search streams. Their capabilities are limited in term of the effectiveness and the efficiency of topic discovery from web search data.

Besides designing effective probabilistic models, exploring efficient training methods is also gaining momentum in recent years. For instance, parallel Gibbs sampling [21] approximated the Gibbs sampling process by synchronous updating of the global distributions. Parallel variational Bayes [38] employed MapReduce to scale the process of parameter inference. [1] presented a scalable parallel framework for efficient inference in latent variable models over streaming web-scale data. [31] described the collapsed variational Bayesian inference for Latent Dirichlet Allocation (LDA) and showed that it is computationally efficient and more accurate than its counterparts. While these parallel techniques demonstrate promising performance, they typically require expensive parallel hardware and the performance-to-price ratio remains unchanged. [36, 37] represented LDA as a factor graph, which enables the classic loopy belief propagation for parameter estimation. [35] proposed an approach to infer topic distribution for new documents in a stream without retraining the model. These parameter interference methods provides useful building blocks for efficient training algorithms of probabilistic models. Hence, it is desirable to develop an efficient parameter inference method that is highly calibred for web search streams.

With the merits of existing probabilistic models and parameter inference methods, none of them tackle the issue of efficiently discovering latent topics from massive web search streams. To the best of our knowledge, this work is the first one that systematically investigates this problem and provides sound solutions. WSSM is able to discover topics from massive web search streams with high accuracy and SPI is able to train WSSM efficiently with superior performance. This work also contributes to many downstream search engine applications which utilize topic modeling to analyze massive web search data.

3. WEB SEARCH STREAM MODEL

As shown in Table 1, search sessions are not explicitly observable in web search streams. Thus, we utilize the method proposed in [14] to identify the search sessions from web search streams. By considering the web search stream of a single user as a doc-

Table 2: Notation

Notation	Description
D	the number of documents
W	the number of query words
U	the number of URLs
K	the number of search topics
d	document
s	search session
z	search topic
w	query word
u	URL
θ	multinomial distribution over topics
ϕ	multinomial distribution over query terms
ψ	multinomial distribution over URLs
α	Dirichlet prior vector for θ
β	Dirichlet prior vector for ϕ
δ	Dirichlet prior vector for ψ
$z^k_{d,s}$	the session s of document d is assigned to topic k
$z^k_{d,s,w}$	the query word w in session s of document d is assigned to topic k
$z^k_{d,s,u}$	the URL u in session s of document d is assigned to topic k
$n_{d,s,w}$	the number of w in session s of document d
$n_{d,s,u}$	the number of u in session s of document d
τ_W	the topically significant threshold of query words
τ_U	the topically significant threshold of URLs
λ_D	the proportion of documents for message passing
λ_K	the proportion of topics for message passing

ument, the web search stream of each user is hierarchically organized as follows: each document contains several search sessions and each search session contains several query words and clicked URLs. To illustrate the underlying logic of the *Web Search Stream Model* (WSSM), we present the notation in Table 2 and describe the generative process of WSSM as follows:

1. for each topic $k \in 1, ..., K$
   (a) draw a query word distribution $\phi_k \sim \text{Dirichlet}(\beta)$;
   (b) draw a URL distribution $\psi_k \sim \text{Dirichlet}(\delta)$;
2. for each document $d \in 1, ..., D$
   (a) draw topic distribution $\theta_d \sim \text{Dirichlet}(\alpha)$;
   (b) for each session s in d
      i. choose a topic $z \sim \text{Multinomial}(\theta_d)$;
      ii. generate query words $w \sim \text{Multinomial}(\phi_z)$;
      iii. generate URLs $u \sim \text{Multinomial}(\psi_z)$ if there exists URL in s;

The assumptions of generating the observed query words and clicked URLs are detailed as follows. When conducting web search, the user first decides the topic that is aligned with his or her current information need and then selects some query words according to the chosen topic to describe the information need. For each search session, the user needs to decide whether to click some URLs to satisfy the information need. The clicked URLs are chosen according to the topic of the corresponding search session as well. We impose the constrain that the query words and the clicked URLs in the same search session should share the same topic, in order to capture semantic coherency of each search session. It is worth mentioning that we do not explicitly create a variable to represent each search query. The underlying reason is that creating variables for search queries involves considerable memory consumption, which heavily impedes the scalability [15]. In fact, we find that utilizing search sessions, query words and URLs in the way defined by WSSM works well in the face of massive web search streams. Essentially, WSSM is a light-weight topic model which captures important ingredients in web search data but avoids complicated relations to facilitate processing massive web search streams.

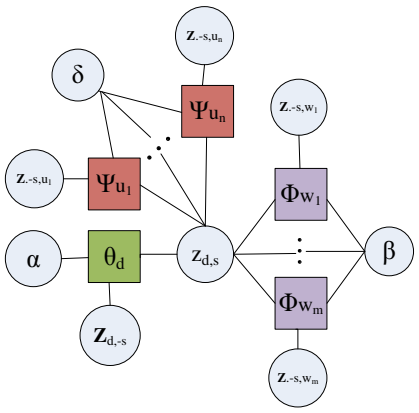


Figure 1: A Fragment of Factor Graph Representation of WSSM. The query word set $\{w_1, ...w_m\}$ and the clicked URL set $\{u_1, ..., u_n\}$ belong to the search session s.

Based on the generative process of WSSM, it is straightforward to design parameter inference methods by collapsed Gibbs sampling (GS) and variational Bayes (VB) [30]. However, in order to achieve better efficiency, we view the topic modeling paradigm of WSSM from a new perspective. We consider the topic modeling task of WSSM as a labeling problem and the objective is to assign a set of topic labels, $\mathbf{z} = \{z^k_{d,s}\}$, to explain the observed data. Figure 1 interprets WSSM by a factor graph. The factors θ_d, $\{\phi_{w_1}, ..., \phi_{w_m}\}$ and $\{\psi_{u_1}, ..., \psi_{u_n}\}$ are denoted by squares. Their connecting variable $z_{d,s}$ is denoted by a circle. The factor θ_d connects topic labels on different search sessions within the same document d, the factor ϕ_w connects topic labels on the query word w but in different search sessions, and the factor ψ_u connects topic labels on the URL u but in different search sessions. Integrating out the parameters $\{\theta, \phi, \psi\}$ based on the Dirichlet-Multinomial conjugacy yields the following joint probability of WSSM:

$$
\begin{aligned}
P(\mathbf{n}, \mathbf{z}|\alpha, \beta, \delta) \propto \\
\prod_{d=1}^{D}\prod_{k=1}^{K} \frac{\Gamma(\sum_{s=1}^{S_d} z^k_{d,s} + \alpha_k)}{\Gamma\Big(\sum_{k=1}^{K}(\sum_{s=1}^{S_d} z^k_{d,s} + \alpha_k)\Big)} \\
\prod_{k=1}^{K}\prod_{w=1}^{W} \frac{\Gamma(\sum_{d=1}^{D}\sum_{s=1}^{S_d} n_{d,s,w} z^k_{d,s,w} + \beta_k)}{\Gamma\Big(\sum_{w=1}^{W}(\sum_{d=1}^{D}\sum_{s=1}^{S_d} n_{d,s,w} z^k_{d,s,w} + \beta_k)\Big)} \\
\prod_{k=1}^{K}\prod_{u=1}^{U} \frac{\Gamma(\sum_{d=1}^{D}\sum_{s=1}^{S_d} n_{d,s,u} z^k_{d,s,u} + \delta_k)}{\Gamma\Big(\sum_{u=1}^{U}(\sum_{d=1}^{D}\sum_{s=1}^{S_d} n_{d,s,u} z^k_{d,s,u} + \delta_k)\Big)} = \\
\prod_{d=1}^{D} f_{\theta_d}(\mathbf{z}_{(d,\cdot)}, \alpha) \prod_{w=1}^{W} f_{\phi_w}(\mathbf{n}_{(\cdot,\cdot,w)}, \mathbf{z}_{(\cdot,\cdot,w)}, \beta) \\
\prod_{u=1}^{U} f_{\psi_u}(\mathbf{n}_{(\cdot,\cdot,u)}, \mathbf{z}_{(\cdot,\cdot,u)}, \delta),
\end{aligned}
\tag{1}
$$

where f_{θ_d}, f_{ϕ_w} and f_{ψ_u} are the factor functions. Specifically, $\mathbf{z}_{(d,\cdot)}$=$\{z_{d,s}, z_{d,-s}\}$, $\mathbf{z}_{\cdot,\cdot,w}$=$\{\mathbf{z}_{d,s,w}, \mathbf{z}_{(\cdot,-s,w)}\}$ and $\mathbf{z}_{\cdot,\cdot,u}$= $\{\mathbf{z}_{d,s,u}, \mathbf{z}_{(\cdot,-s,u)}\}$ denote subsets of the variables in Figure 1.

4. STREAM PARAMETER INFERENCE

In this section, we discuss the details of the *Stream Parameter Inference* (SPI). Section 4.1 presents the *Web Search Matrix* (WSM) that is utilized to select the topically significant query words and URLs from massive web search streams. Section 4.2 shows the procedure of utilizing SPI to train WSSM with web search data in a specific time period. Section 4.3 discusses the approach of applying SPI across a range of time periods.

4.1 Web Search Matrix

Given the massive size of web search streams and the demanding requirement of efficiency in real-life search engine applications, it is impractical to analyze every detail of these streams with WSSM. WSSM relies on the co-occurrences of query words and URLs to compose latent topics. Hence, the query words and URLs whose occurrences are significant count for the major contents of latent topics. In order to speed up the training process of WSSM, we define the *Web Search Matrix* (WSM), which extends methods proposed for capturing stream statistics [9] and helps to identify query words and URLs that have potential to become the major contents of latent topics. WSM works as an upstream subroutine of SPI and it enjoys the advantages of efficient update and small memory consumption.

A WSM denoted by A is represented by a two-dimensional array with parameters (ε, l_1, l_2), where l_1 is the number of rows and l_2 is the number of columns. Each entry in A is initialized as zero and l_1 hash functions are chosen uniformly at random from a pairwise independent family. The parameter $\varepsilon \in (0, 1)$ determines the proportion of rows allocated for storing the information of query words, i.e., $\{A[1,1], ..., A[\varepsilon l_1, l_2]\}$ are utilized for storing the information of query words while $\{A[\varepsilon l_1 + 1, 1], ..., A[l_1, l_2]\}$ are utilized for storing the information of URLs. When a new entry of web search data arrives, the search query is tokenized into query words. For each query word w, its quantity c_w is added to an entry in each row that is less or equal to εl_1 and the position of the entry is determined by the corresponding hash function. Specifically, $\forall j, 1 \leq j \leq \varepsilon l_1$,

$$A[j, h_j(w)] = A[j, h_j(w)] + c_w. \tag{2}$$

Similarly, the clicked URL u is updated for $\forall j, \varepsilon l_1 + 1 \leq j' \leq l_1$ as follows:

$$A[j', h_{j'}(u)] = A[j', h_{j'}(u)] + c_u. \tag{3}$$

For each WSM, we create a query-word heap and a URL heap to store the topically significant query words and URLs. The frequency of a query word w can be estimated by

$$min_{\forall j, 1 \leq j \leq \varepsilon l_1} A[j, h_j(w)].$$

If the estimation is above the threshold of $\tau_W W$, the query word w is added to the query-word heap. The heap checks whether the estimation for the word with lowest count is above threshold; if not, it is deleted from the heap. The frequency of each query word can be estimated by using time $O(\varepsilon l_1)$. Every query word that occurs more than $\tau_W W$ times is identified as topically significant query word, and with probability $(1 - e^{-\varepsilon l_1})$, no query word whose frequency is less than $(\tau_W - \frac{e}{l_2})W$ is identified as topically significant query word. Similarly, the frequency of a URL u is estimated by

$$min_{\forall j, \varepsilon l_1 + 1 \leq j \leq l_1} A[j, h_j(u)]$$

. If the estimation of u is above the threshold of $\tau_U U$, it is added to the URL heap. The frequency of a URL can be estimated from a web search stream by using time $O((1 - \varepsilon) l_1)$. Every URL that occurs more than $\tau_U U$ times is identified as topically significant URL, and with probability $(1 - e^{-(1-\varepsilon) l_1})$, no URL whose count is less than $(\tau_U - \frac{e}{l_2})U$ is identified as topically significant URL.

We proceed to prove the above arguments by focusing on query words. The proof corresponding to URLs can be straightforwardly obtained in a similar approach and it is skipped due to space limitation. We assign the indicator $I_{i,j,k}$ to 1 if $(i \neq k)$ and $h_j(i) = h_j(k)$. According to pairwise independence of the hash functions, the expectation of $I_{i,j,k}$ is as follows:

$$E(I_{i,j,k}) \leq \frac{1}{l_2}. \tag{4}$$

Since $A[j, h_j(i)] = frequency(w_i) + \sum_{k=1}^{n} I_{i,j,k} frequency(w_k)$, we then obtain the following inequality:

$$\begin{aligned} P[\forall_j, A[j, h_j(i)] > frequency(w_i) + \frac{e}{l_2} W] = \\ P[\forall_j, frequency(w_i) + \sum_{k=1}^{n} I_{i,j,k} frequency(w_k) > \\ frequency(w_i) + \frac{e}{l_2} W] = \\ P[\forall_j, \sum_{k=1}^{n} I_{i,j,k} frequency(w_k) > \frac{e}{l_2} W] < e^{-\varepsilon l_1}. \end{aligned} \tag{5}$$

Since the threshold is ever increasing, no topically significant query word is omitted by checking estimated frequency when the frequency of a query word increases. The probability of mistakenly outputting a query word whose frequency is less than $(\tau_W - \frac{e}{l_2})W$ is bounded by $(1 - e^{-\varepsilon l_1})$. The above discussion verifies the effectiveness of WSM in efficiently discovering the topically significant query words and URLs. The query-word heap and the URL heap are represented in a space-efficient way by using two arrays. When utilizing the WSM technique in SPI, we create a WSM instance for each web search stream that corresponds to each search engine user as well as a WSM instance for storing the global information of all web search streams. In the downstream subroutines of SPI, we only utilize the query words and URLs that are topically significant in either the user's WSM or the global WSM for parameter inference. In this way, these subroutines handle much less web search data and thus significant speedup performance can be achieved.

4.2 Parameter Inference for A Time Period

The above subsection describes the method of selecting the topically significant query words and URLs. In this subsection, we discuss how to conduct latent parameter inference based on the selected query words and URLs within a time period. For topic modeling purpose, we need to calculate the following probability:

$$P(z^k_{d,s} | \mathbf{z}^k_{d,-s}, \mathbf{n}_{d,-s}, \mathbf{z}^k_{\cdot,-s,w}, \mathbf{n}_{\cdot,-s,w}, \mathbf{z}^k_{\cdot,-s,u}, \mathbf{n}_{\cdot,-s,u}), \tag{6}$$

where $-s$ denotes all session indices except s, and $\mathbf{z}_{d,-s}$, $\mathbf{z}_{\cdot,-s,w}$ and $\mathbf{z}_{\cdot,-s,u}$ represent all possible topic assignments of neighboring variables. The above probability is referred as the message $\mu^k_{d,s}$. Using the Bayes' rule and the joint probability of Equation (1), we expand Equation (6) and obtain the following message update formula:

$$\mu_{d,s}^k \propto \frac{p(\mathbf{z}_{d,\cdot}^k, \mathbf{n}_{d,\cdot})}{p(\mathbf{z}_{d,-s}^k, \mathbf{n}_{d,-s})} \prod_{w \in s} \frac{p(\mathbf{z}_{\cdot,\cdot,w}^k, \mathbf{n}_{\cdot,\cdot,w})}{p(\mathbf{z}_{\cdot,-s,w}^k, \mathbf{n}_{\cdot,-s,w})} \prod_{u \in s} \frac{p(\mathbf{z}_{\cdot,\cdot,u}^k, \mathbf{n}_{\cdot,\cdot,u})}{p(\mathbf{z}_{\cdot,-s,u}^k, \mathbf{n}_{\cdot,-s,u})}$$
$$\propto \frac{\mu_{d,-s}^k + \alpha_k}{\sum_{k'} \left(\mu_{d,-s}^{k'} + \alpha_{k'}\right)} \frac{\Gamma\Big(\sum_{w'}(n_{\cdot,-s,w'}\mu_{\cdot,-s,w'}^k + \beta_{w'})\Big)}{\Gamma\Big(\sum_{w'}(n_{\cdot,-s,w'}\mu_{\cdot,-s,w'}^k + \beta_{w'} + n_{d,s,w'})\Big)} \prod_{w \in s} \Big(\frac{\Gamma(n_{\cdot,-s,w}\mu_{\cdot,-s,w}^k + \beta_w + n_{d,s,w})}{\Gamma(n_{\cdot,-s,w}\mu_{\cdot,-s,w}^k + \beta_w)}\Big)$$
$$\frac{\Gamma\Big(\sum_{u'}(n_{\cdot,-s,u'}\mu_{\cdot,-s,u'}^k + \beta_{u'})\Big)}{\Gamma\Big(\sum_{u'}(n_{\cdot,-s,u'}\mu_{\cdot,-s,u'}^k + \beta_{u'} + n_{d,s,u'})\Big)} \prod_{u \in s} \Big(\frac{\Gamma(n_{\cdot,-s,u}\mu_{\cdot,-s,u}^k + \beta_u + n_{d,s,u})}{\Gamma(n_{\cdot,-s,u}\mu_{\cdot,-s,u}^k + \beta_u)}\Big). \tag{7}$$

After updating the messages, we normalize them by their corresponding normalization factors. Then $\mu_{d,s,w}^k$ and $\mu_{d,s,u}^k$, i.e., the messages corresponding to the query word w and the URL u in the session s are updated as follows:

$$\mu_{d,s,w}^k = \mu_{d,s,u}^k = \mu_{d,s}^k. \tag{8}$$

The messages are passed from variables to factors, and in turn from factors to variables for a predefined number of iterations or convergence is achieved. It is worth mentioning that we only need to pass messages whose corresponding $n_{d,s,w}$ or $n_{d,s,u}$ are not zero. As $\mathbf{n}$ is very sparse in practice, Equation (7) is computationally efficient by sweeping only nonzero elements in $\mathbf{n}$.

To efficiently train WSSM, we select a fraction of query words, URLs, documents and topics for fast topic discovery. The basic idea is to choose only the topically significant query words and URLs, i.e., for each document, we select query words and URLs which are either topically significant according to the global WSM or topically significant in terms of the corresponding user's WSM. Then we select the best message update order based on the message residuals $\{r_{d,s}^k\}$, which is the absolute difference between two messages at successive iterations t and $(t-1)$:

$$r_{d,s}^k = \Big|\mu_{d,s}^k(t) - \mu_{d,s}^k(t-1)\Big|. \tag{9}$$

By sequentially updating messages in a descending order of the message residuals in each iteration, SPI converges faster to a fixed point than those without considering the sorted order. The reason is that updating messages with top largest residuals efficiently accelerates the speed of convergence and the updated messages with the largest residuals in turn influence their neighboring messages, which contribute to fast convergence as well. After sorting the message residuals, in each iteration we select a fraction of documents for message updating and passing. For each document, we search a fraction of topic space for message updating and passing. Sorting the residuals in Equation (9) is computationally expensive because the number of non-zero residuals is very large. In practice, we turn to sorting the accumulated residuals at topics. We define the residual of a specific document d at topic k as follows:

$$r_d^k = \sum_s r_{d,s}^k. \tag{10}$$

Then the residual of the document d is calculated as follows:

$$r_d = \sum_k r_d^k. \tag{11}$$

After each iteration, we sort r_d in a descending order for all documents and select $\lambda_D D$ documents with the largest residuals r_d. We then sort r_d^k in a descending order for each selected document, and select the subset topics $\lambda_K K$ with the largest residuals r_d^k. In the following iteration, we update messages for the subset documents $\lambda_D D$ and the subset topics $\lambda_K K$, and thus save enormous time in each iteration. At the first iteration, SPI scans the whole document space and the topic space. In the remaining iterations, based on residuals, SPI actively selects $\lambda_K K$ topics for message updating and passing. According to the selected $\lambda_K K$ topics, we normalize the local messages as follows:

$$\hat{\mu}_{d,s}^k(t) = \frac{\mu_{d,s}^k(t)}{\sum_k \mu_{d,s}^k(t)} \times \sum_k \hat{\mu}_{d,s}^k(t-1), \tag{12}$$

$$\hat{\mu}_{d,s,w}^k(t) = \frac{\mu_{d,s,w}^k(t)}{\sum_k \mu_{d,s,w}^k(t)} \times \sum_k \hat{\mu}_{d,s,w}^k(t-1), \tag{13}$$

$$\hat{\mu}_{d,s,u}^k(t) = \frac{\mu_{d,s,u}^k(t)}{\sum_k \mu_{d,s,u}^k(t)} \times \sum_k \hat{\mu}_{d,s,u}^k(t-1), \tag{14}$$

where $\hat{\mu}_{d,s}^k(t-1)$, $\hat{\mu}_{d,s,w}^k(t-1)$ and $\hat{\mu}_{d,s,u}^k(t-1)$ are the normalized messages in the previous iteration, $\hat{\mu}_{d,s}^k(t)$, $\hat{\mu}_{d,s,w}^k(t)$ and $\hat{\mu}_{d,s,u}^k(t)$ are the normalized messages in the current iteration. At negligible computational cost, the sorted residuals can be resorted during message passing. If the residuals are in almost sorted order, only a few swaps will restore the sorted order by the standard insertion sort [22] and a lot of sorting time will be saved. Based on the inferred messages, we estimate the latent parameters θ, ϕ and ψ as follows:

$$\theta_{dk} = \frac{\mu_{d,\cdot}^k + \alpha_k}{\sum_{k'} \left(\mu_{d,\cdot}^{k'} + \alpha_{k'}\right)}. \tag{15}$$

$$\phi_{kw} = \frac{\mu_{\cdot,\cdot,w}^k + \beta_w}{\sum_{w'} \left(\mu_{\cdot,\cdot,w'}^k + \beta_{w'}\right)}. \tag{16}$$

$$\psi_{ku} = \frac{\mu_{\cdot,\cdot,u}^k + \delta_u}{\sum_{u'} \left(\mu_{\cdot,\cdot,u'}^k + \delta_{u'}\right)}. \tag{17}$$

The technical details of SPI are presented in Algorithm 1. Query words and URLs are selected for topic modeling if they are in the global WSM instance or the user-specific WSM instance. At the first iteration, SPI scans the entire corpus of web search data within the time period, searches the complete topic space, computes and sorts residuals. For the remaining iterations, SPI actively selects the subset documents $\lambda_D D$ and the subset topics $\lambda_K K$ for message updating and passing. After each iteration, SPI dynamically refines and sorts residuals. SPI terminates when the convergence condition is satisfied or the maximum iteration number is reached.

Algorithm 1 Stream Parameter Inference (SPI)

for each session s in each document d and each topic k **do**
  $\mu^k_{d,s}(0) \leftarrow$ random initialization and normalization;
  for query word w in the global WSM query-word heap or in the user's WSM query-word heap **do**
    initialize $\mu^k_{d,s,w}(0)$ according to $\mu^k_{d,s}(0)$;
  end for
  for URL u in the global WSM URL heap or in the user's WSM URL heap **do**
    initialize $\mu^k_{d,s,u}(0)$ according to $\mu^k_{d,s}(0)$;
  end for
end for
for $d \in [1, D]$ **do**
  for $k \in [1, K]$ **do**
    compute $\mu^k_{d,s}(1)$ by Equation (7);
    compute $\mu^k_{d,s,w}(1)$, $\mu^k_{d,s,u}(1)$;
    compute $r^k_{d,s}(1)$;
  end for
  sort $r^k_d(1)$ by descending order;
  select the top $\lambda_K K$ topics;
end for
sort $r_d(1)$ by descending order;
select the top $\lambda_D D$ documents;
for $t \leftarrow 2$ to T **do**
  for $d \in [1, \lambda_D D]$ **do**
    for $k \in [1, \lambda_K K]$ **do**
      compute $\mu^k_{d,s}(t)$ by Equation (7);
      compute $\mu^k_{d,s,w}(t)$, $\mu^k_{d,s,u}(t)$;
      compute $r^k_{d,s}(t)$;
    end for
    sort $r^k_d(t)$ by descending order;
    select the top $\lambda_K K$ topics;
  end for
  sort $r_d(t)$ by descending order;
  select the top $\lambda_D D$ documents;
end for

4.3 Parameter Inference for Time Periods

Algorithm 1 shows how to train WSSM by utilizing web search data within a specific time period. In practice, web search data comes in the form of streams. We can adapt Algorithm 1 into a life-long algorithm that is able to process online web search steams. Based on the timestamps of web search entries, we divide web search stream data into several time periods. Hence, SPI takes the input as a series of time periods, $w \in [1, \infty)$, $u \in [1, \infty)$, $b \in [1, \infty)$, $d \in [1, D_b]$. b is the index of the period and D_b the number of documents within the period. Note that the time period index b, the word index w and the URL index u can reach infinity in order to account for infinite web search data. In order to capture the phenomenon of topic evolution, we update the parameters θ, ϕ and ψ based on the information of the previous and the current time periods. We update ϕ and ψ dynamically. For each time period, SPI randomly initializes the messages $\mu^k_{d,s}(b)$ and initializes the sufficient statistics of the previous time period as follows:

$$\Delta_{k,w} = n_{\cdot,\cdot,w}(b-1)\mu^k_{\cdot,\cdot,w}(b-1). \tag{18}$$

$$\Delta_{k,u} = n_{\cdot,\cdot,u}(b-1)\mu^k_{\cdot,\cdot,u}(b-1). \tag{19}$$

Then the message updating formula is adapted as follows:

$$\mu^k_{d,s} \propto \frac{\mu^k_{d,-s}(b) + \alpha_k}{\sum_{k'}\left(\mu^{k'}_{d,-s}(b) + \alpha_{k'}\right)} \cdot \frac{\Gamma\Big(\sum_{w'}(n_{\cdot,-s,w'}(b)\mu^k_{\cdot,-s,w'}(b) + \Delta_{k,w'} + \beta_{w'})\Big)}{\Gamma\Big(\sum_{w'}(n_{\cdot,-s,w'}(b)\mu^k_{\cdot,-s,w'}(b) + \Delta_{k,w'} + \beta_{w'} + n_{d,s,w'})\Big)} \prod_{w\in s}\Big(\frac{\Gamma(n_{\cdot,-s,w}(b)\mu^k_{\cdot,-s,w}(b) + +\Delta_{k,w} + \beta_w + n_{d,s,w})}{\Gamma(n_{\cdot,-s,w}(b)\mu^k_{\cdot,-s,w}(b) + \Delta_{k,w} + \beta_w)}\Big) \cdot \frac{\Gamma\Big(\sum_{u'}(n_{\cdot,-s,u'}(b)\mu^k_{\cdot,-s,u'}(b) + \Delta_{k,u'} + \beta_{u'})\Big)}{\Gamma\Big(\sum_{u'}(n_{\cdot,-s,u'}(b)\mu^k_{\cdot,-s,u'}(b) + \Delta_{k,u'} + \beta_{u'} + n_{d,s,u'})\Big)} \prod_{u\in s}\Big(\frac{\Gamma(n_{\cdot,-s,u}(b)\mu^k_{\cdot,-s,u}(b) + \Delta_{k,u} + \beta_u + n_{d,s,u})}{\Gamma(n_{\cdot,-s,u}(b)\mu^k_{\cdot,-s,u}(b) + \Delta_{k,u} + \beta_u)}\Big). \tag{20}$$

With the new message updating formula, we run Algorithm 1 on the web search data in the current time period until convergence or the maximum iteration number is reached. To make efficient I/O from disk to memory, we load frequently visited entries of Δ in buffer to reduce reading and writing frequently visited statistics of the previous time period. When a new query word or a new URL is identified, we increment the vocabulary size by one. Incrementing vocabulary size implies that the distribution ϕ and ψ are generated by a Dirichlet distribution with increasing dimensions. However, it does not change message updating very much when W and U are large. In practice, the aforementioned heuristics of incrementing vocabulary size works quite well for web search streams.

5. EXPERIMENTS

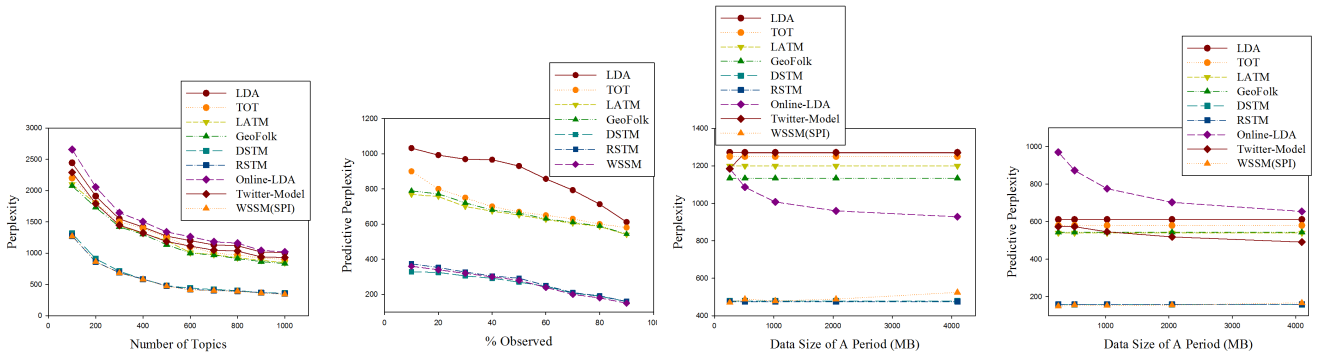
In this section, we evaluate the performance of WSSM and SPI with a large-scale search engine query log. Section 5.1 describes the experimental setup. Section 5.2 quantitatively evaluates WSSM with several standard metrics. Section 5.3 presents the experimental results of SPI in terms of speedup performance. Section 5.4 gauges the memory consumption of SPI. Section 5.5 reports some discovered topics and analyzes the phenomenon of topic evolution.

5.1 Experimental Setup

We utilize a three-month query log from a commercial search engine to prepare the experimental data. The query log contains approximately 53 million search sessions over 1,812,942 distinct web search entries of 2,381,345 users. We fix the hyperparameters as $\alpha = 2/K$, $\beta = 0.01$ and $\delta = 0.01$ like [24] and this setting demonstrates fairly good empirical performance in our experiments. For sampling based parameter inference methods, we report the topic modeling results after 5000 iterations, which practically ensures convergence in terms of perplexity that is a standard measure for evaluating the generalization of a probabilistic model [26]. Unless otherwise mentioned, when applying SPI to train WSSM, we set λ_D and λ_K to 0.5. The aforementioned parameters strike a good balance between efficiency and accuracy.

5.2 Quantitative Measures

A held-out dataset containing about eight thousand web search stream entries is utilized to evaluate WSSM's capability of predicting unseen query words in terms of perplexity [26]. Perplexity is monotonically decreasing in the likelihood of the held-out data. A lower perplexity indicates better generalization performance. Specifically, perplexity is calculated by the following equation:



(a) Perplexity Comparison (b) Predictive Perplexity Comparison (c) Perplexity With Different Data Sizes of A Time Period (d) Predictive Perplexity With Different Data Sizes of A Time Period

Figure 2: Perplexity Comparison of Different Probabilistic Topic Models

$$Perplexity_{held-out}(\mathcal{M}) = (\prod_{d=1}^{D}\prod_{i=1}^{N_d} p(w_i|\mathcal{M}))^{\frac{-1}{\sum_{d=1}^{D}(N_d)}}, \quad (21)$$

where $\mathcal{M}$ is the model learned from the training process. The baselines we choose for this comparative experiment are six retrospective topic models (i.e., LDA [24], TOT [34], LATM [32], GeoFolk [28], DSTM and RSTM [15]) and two topic models proposed for stream data (i.e., Online-LDA [12] and Twitter-Model [13]). The parameter inference methods of these models are those described in the corresponding papers. In this experiment, for Online-LDA, Twitter-Model and WSSM(SPI), we consider the web search data of each day as a period. The result is presented in Figure 2(a), from which we observe that WSSM demonstrates good capability in predicting unseen data comparing with the baselines. For example, when the number of latent topics is set to 1000, the perplexity of LDA is 1009, that of TOT is 902, that of LATM is 847, that of GeoFolk is 829. Compared with the other models, DSTM, RSTM and WSSM(SPI) significantly reduces the perplexity and achieves a perplexity around 360. Although WSSM achieves similar performance as the state-of-the-art retrospective query log topic models such as DSTM and RSTM, we will show that it consumes significantly less time than the two counterparts.

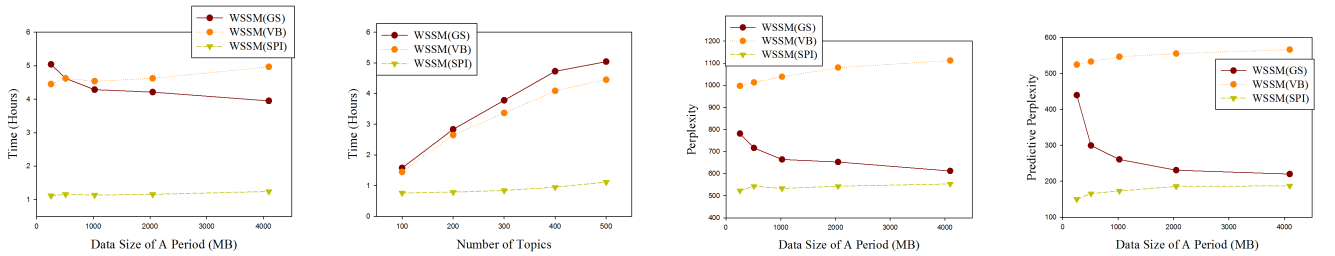
The second metric aims to gauge how effective the proposed models are in predicting the remaining query words after observing a portion of the user's web search history. Suppose we observe the query words $w_{1:P}$ from a user's query log, we are interested in finding which model provides a better predictive distribution $p(w|w_{1:P})$ of the remaining query words. Specifically, we use each user's eighty percent of search queries as the training data and the remaining twenty percent as the testing data. We use Equation (22) to calculate the perplexity of the testing data. The comparison results are presented in Figure 2(b). We observe that WSSM(SPI) demonstrates good capability to predicting the user's future web search data given the user's search history. When the topic is set to 500 and 90% of the web search data is set to be observed. LDA demonstrates a perplexity of 621, LATM generates a perplexity of 551 and GeoFolk shows a perplexity of 554. WSSM(SPI) demonstrates a perplexity of 152. This result shows that WSSM(SPI) has a good capability of predicting the user's web search given the user's web search history.

$$Perplexity_{portion}(\mathcal{M}) = (\prod_{d=1}^{D}\prod_{i=P+1}^{N_d} p(w_i|\mathcal{M}, w_{a:P}))^{\frac{-1}{\sum_{d=1}^{D}(N_d-P)}}. \quad (22)$$

Figures 2(c) and 2(d) show the held-out perplexity and predictive perplexity with the increase of the data size of a period. Figure 2(c) shows the held-out perplexity when the topic amount is fixed to 500. All the retrospective topic models show stable perplexity values since they can only be trained in an offline fashion. Hence, the held-out perplexity of the retrospective models is not sensitive to the change of the data size of a period. Online-LDA demonstrates a lower perplexity when the data size of a period increases because larger data size leads to better online gradient descents for higher topic modeling accuracy. Although WSSM(SPI) performs slightly worse when the data size of a period increases, WSSM(SPI) achieves high topic modeling accuracy. Figure 2(d) shows the predictive perplexity with the increase of the data size of a period when K is set to 500 and 90% of the web search data is observed. Again, all the retrospective topic models show stable predictive perplexity. Online-LDA and Twitter-Model demonstrate lower perplexity as the data size of a period increases while the predictive perplexity of WSSM(SPI) goes slightly higher when the data size of a period increases. However, WSSM(SPI) still maintains superior performance. These experimental results verify that WSSM is a robust and effective topic model for web search streams in terms of the topic modeling accuracy.

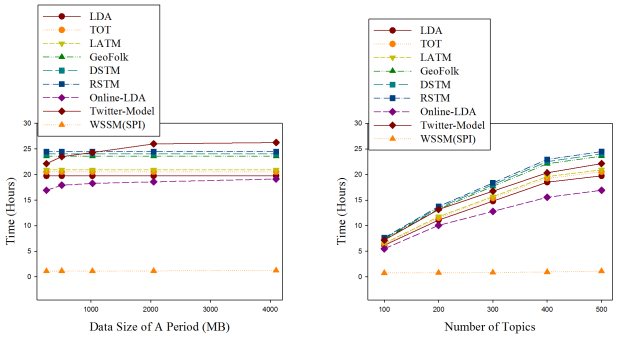
5.3 Speedup Performance

We proceed to gauge the efficiency of SPI and verify the speedup effect of SPI by comparing it with other alternatives such as variational Bayes [2] (VB) and collapsed Gibbs sampling (GS) [11, 24]. In order to make fair comparison with the two counterparts, we choose to train them all on the topically significant query words and URLs that are obtained from WSM. Figure 3(a) shows the training time with the increase of the data size of a period when K is set to 500. The training time of WSSM(VB) and WSSM(SPI) increases with the data size of a period, while the training time of WSSM(GS) slightly decreases with the data size of a period. The reason is that sampling based parameter inference methods converge slightly faster at relatively larger data size of a period. WSSM(SPI) runs much faster than the other two baselines even it involves the additional cost of residual sorting. We observe that WSSM(SPI) not only consumes the least training time but also is



(a) Consumed Time With Different Data Sizes of A Time Period (b) Consumed Time With Different Topic Amounts (c) Perplexity With Different Data Sizes of A Time Period (d) Predictive Perplexity With Different Data Sizes of A Time Period

Figure 3: Speedup Evaluation of WSSM Trained by Different Parameter Inference Methods



(a) Time With Different Data Sizes of A Period (b) Time With Different Topic Amounts

Figure 4: Speedup Comparison With Other Topic Models

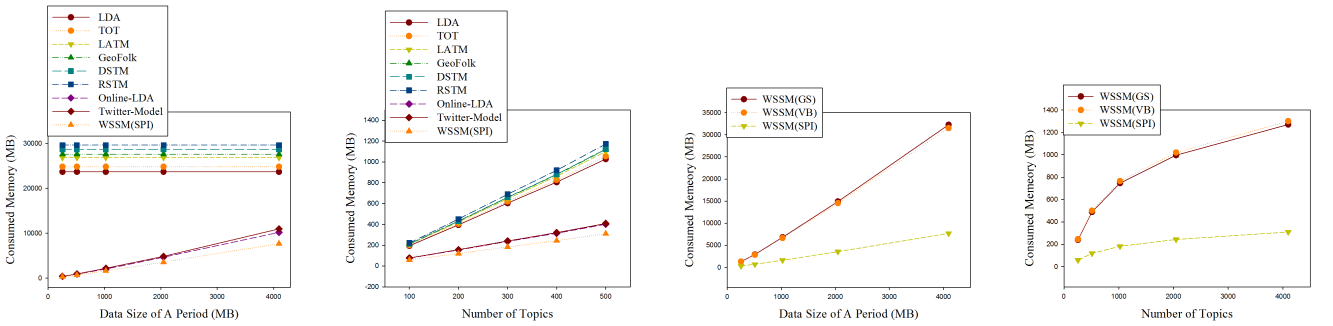
the least insensitive to the change of the data size of a period. The experimental result confirms that the proposed SPI algorithm can work faster than the other two algorithms while retains a good accuracy in topic modeling of massive web search data. Figure 3(b) shows the consumed time of the three parameter inference methods as the topic amount K increases and the data size of a period is fixed at 256MB. The training time of all the three parameter inference algorithms increases with K, since a larger topic space needs to be scanned. We observe that the training time of GS and VB is close to SPI when K is small but is much longer than that of SPI when K becomes large. The major reason is that the other two parameter inference algorithms require visiting all documents and the entire topic space, while SPI selectively chooses a subset of documents and a fraction of topic space. This result shows that GS and VB are not promising candidates for fast topic modeling of massive web search streams due to the tedious scanning process while SPI can effectively avoid this drawback.

Figure 3(c) shows the held-out perplexity with increasing the data size of a time period when K is fixed at 500. WSSM(VB) demonstrates lower perplexity when the data size of a period increases, because larger data size of period leads to more robust online gradient descents for higher accuracy. However, WSSM(GS) and WSSM(SPI) often perform worse when the data size of a period increases, because smaller data size of a period helps correct the global biases. In all cases, WSSM(SPI) achieves the lowest predictive perplexity showing the highest topic modeling accuracy. Figure 3(d) shows the predictive perplexity with the increase of the data size of a period when topic amount is fixed to 500 and 90% web search data is observed. We find that this experimental result is similar to that in Figure 3(c). WSSM(GS) demonstrates lower perplexity when the data size of a period increases. However, WSSM(VB) and WSSM(SPI) often perform worse when the data size of a period becomes larger. WSSM(SPI) achieves the lowest predictive perplexity, showing that SPI keeps the highest topic modeling accuracy with different data sizes of a period.

The above experimental results show that SPI is a promising method for training WSSM. A natural question arises that whether WSSM(SPI) outperforms the other topic models in terms of training efficiency. Figures 4(a) and 4(b) show the comparative results of the time consumption of different models when the data size of a period increases and the topic amount increases. In this evaluation, all the baselines are trained on full web search data while WSSM(SPI) is trained on the topically significant query words and URLs. Figure 4(a) shows the training time of each topic model when the topic amount is set to 500. The result demonstrates that the efficiency of WSSM(SPI) is much better than the other topic models. The underlying reasons are essentially threefold. First, WSSM is a relatively light-weight topic model and does not involve much complicated calculation. Second, WSSM(SPI) utilizes WSM to reduce the web search data that need to be digested by the downstream topic modeling process. Third, WSSM(SPI) reduces the amount of documents and the scope of the topic space that need to be scanned in each iteration. Figure 4(b) shows the consumed time with the increase of the topic amount K when the data size of a period is fixed at 256MB. The consumed time of all the involved topic models is roughly liner in the number of topics. However, the time increase of WSSM(SPI) is rather moderate and it demonstrates significantly better scalability when the number of topics is large.

5.4 Memory Consumption

In this subsection, we evaluate the memory consumptions of the proposed techniques with about ten gigabytes web search data. Figure 5(a) shows the memory consumption of different probabilistic topic models when we increase the data size of a period. The retrospective topic models such as LDA need to process all the data together and typically consume memory that is several times of the size of the entire data. In contrast, the other three topic models can take better advantage of the small data size of each period and consume significantly less memory. This phenomenon shows that the topic models that are specialized for streams are more capable of processing large-scale web search data. For example, when the data size of a period is set to 256MB, WSSM(SPI) typically consumes about 310MB memory, which is much less than those consumed by the retrospective topic models. Even compared with Online-LDA and Twitter-Model, WSSM(SPI) also keeps the superiority in terms of memory consumption. This result demonstrates that SPI can significantly reduce the memory consumption by utilizing WSM. Fig-



(a) Memory Consumption With Different Data Sizes of A Period (Other Models)

(b) Memory Consumption With Different Topic Amounts (Other Models)

(c) Memory Consumption With Different Data Sizes of A Period (VB, GS, SPI)

(d) Memory Consumption With Different Topic Amounts (VB, GS, SPI)

Figure 5: Performance Comparison in Terms of Memory

ure 5(b) shows the memory consumption of different topic models when we fix the data size of a period at 256MB and increase the topic amount K. The memory consumption of each topic model increases linearly with the number of topics. However, Online-LDA, Twitter-Model and WSSM(SPI) demonstrate the lowest consumed memory when topic amount becomes larger, showing their superior scalability in handling a large number of latent topics. Figures 5(c) and 5(d) compare the memory consumption of different parameter inference methods for training WSSM. The SPI algorithm always consumes less memory than both GS and VB. These results again verify the effectiveness of WSM, which only selects the topically significant query words and URLs for downstream topic modeling. Thus, SPI only needs to process a subset of the web search data that is digested by the other two parameter inference methods.

5.5 Discovered Topics and Topic Evolution

The above subsections qualitatively gauge the performance of WSSM and SPI. An informal but important measure of the success of topic models is the plausibility of the discovered topics. By analyzing the topic modeling results of WSSM, we observe that that WSSM is able to obtain semantically meaningful topics by different parameter inference methods. Table 3 shows the top ten words of four topics extracted by VB, GS and SPI on the same dataset. We see that all the three parameter inference methods can effectively group semantically coherent query words together as topics, where most of the top ten words are similar except the slightly different word ranking. For example, in the topic *Vehicle*, the topics discovered by the three parameter inference methods all contain query words such as "car", "wheel" "recall", "engine" and "hybrid". The rankings of these query words are also similar across the topics discovered by the three different methods. Based on these empirical results, we find that the discovered topics are comparable among all the three parameter inference algorithms. These results empirically verify that utilizing SPI to train WSSM can achieve significantly better efficiency than GS and VB without sacrificing topic modeling accuracy.

Besides effectively discovering latent topics from massive web search streams, we can also identify the evolution of each topic by comparing the topics that are discovered from different time periods. Table 4 shows an example (i.e., the topic *Vehicle*) of topic evolution detected by the SPI algorithm. In this first time period, the query word "car" is not in the top five words. As the web search streams flow, in the second time period, the rank of "car" becomes the fourth. In the third, fourth and fifth time periods, the query word "car" is always ranked as top one in this topic. Similarly, the query words "price" and "white"gradually become more and more important in the topic of *Vehicle*. Besides capturing the dynamic shift of different word rankings, we can also observe the phenomenon of word emergence and word perishment. In the fourth period, the word "nissan" first appears in the topic of *Vehicle* and its rank increases in the fifth period. On the other hand, the word "lexus" becomes less important as more web search data is observed and "lexus" totally disappears in the third period. This empirical result shows that training WSSM via SPI with web search data of different time periods is able to detect the topic evolution over time. The topic evolution provides a window of observing general web dynamics and popularity of different entities on the web.

6. CONCLUSIONS

In this paper, we study the issue of efficiently discovering latent topics from massive web search streams. We propose the *Web Search Stream Model* and the *Stream Parameter Inference* , which are critical for fast topic discovery from massive web search streams. We conduct extensive experiments based on a large-scale search engine query log in order to gauge the performance of the proposed techniques. The experimental results show that WSSM is effective to discover latent topics from voluminous web search streams and SPI significantly accelerates the process of topic modeling while keeping a superior topic modeling accuracy. In addition, we also show that WSSM and SPI are able to reduce memory consumption and capture the phenomenon of topic evolution as the web search streams flow. In future work, a promising direction is to explore more potential applications of the proposed techniques and investigate whether they can be transferred to related scenarios such as microblog data analysis.

7. ACKNOWLEDGMENTS

This work is partially supported by GRF under grant numbers HKUST 617610. We thank the anonymous reviewers and area chairs for their constructive comments.

8. REFERENCES

[1] Amr Ahmed, Moahmed Aly, Joseph Gonzalez, Shravan Narayanamurthy, and Alexander J Smola, *Scalable inference in latent variable models*, WSDM, 2012.

[2] Hagai Attias, *Inferring parameters and structure of latent variable models by variational bayes*, UAI, 1999.

[3] Ricardo Baeza-Yates and Yoelle Maarek, *(big) usage data in web search*, WSDM, 2013.

Table 3: Topics Discovered via WSSM by Different Methods. The name of each topic is the authors' interpretation.

	Vehicle		**Music**		**Hotel**		**Food**	
Collapsed Gibbs Sampling	car	0.038868	lyrics	0.011074	hotel	0.058742	kitchen	0.032003
	wheel	0.036804	music	0.010603	inn	0.050484	chicken	0.028573
	white	0.035446	band	0.010385	city	0.045403	mcdold	0.024160
	toyota	0.031700	tennessee	0.007768	resort	0.032232	fish	0.022939
	seat	0.023967	rhythm	0.007688	motel	0.020342	sauce	0.020757
	engine	0.021543	team	0.005291	suite	0.016651	price	0.019469
	price	0.010062	genres	0.004625	time	0.013107	recipe	0.010877
	lexus	0.009370	opera	0.003653	hyatt	0.009001	pasta	0.006485
	recall	0.007322	westlife	0.003653	sheraton	0.006221	homemade	0.004687
	hybrid	0.004487	musician	0.003187	plaza	0.004502	beef	0.004687
	Vehicle		**Music**		**Hotel**		**Food**	
Varitional Bayes	car	0.030495	lyrics	0.011155	hotel	0.059773	kitchen	0.031739
	wheel	0.029165	music	0.010921	inn	0.048274	chicken	0.030023
	nissan	0.027058	tennessee	0.009005	motel	0.036013	mcdold	0.020025
	toyota	0.017009	singer	0.008446	resort	0.034995	fish	0.019800
	seat	0.016980	rhythm	0.007705	reserve	0.028452	burger	0.018905
	engine	0.013403	team	0.005306	suite	0.024337	price	0.012428
	honda	0.012715	genres	0.005252	time	0.022912	cooker	0.011694
	lexus	0.009101	opera	0.004769	hyatt	0.015959	pasta	0.009845
	recall	0.008984	westlife	0.003571	sheraton	0.015959	beef	0.008647
	hybrid	0.004866	musician	0.003113	plaza	0.005913	cookie	0.004369
	Vehicle		**Music**		**Hotel**		**Food**	
Stream Parameter Inference	car	0.035480	lyrics	0.010938	hotel	0.056963	kitchen	0.030791
	price	0.034921	music	0.009368	inn	0.045522	chicken	0.030405
	white	0.028533	band	0.008245	city	0.038016	mcdold	0.029897
	toyota	0.027191	style	0.007059	resort	0.035438	fish	0.021859
	brake	0.026614	rhythm	0.007059	sheraton	0.029773	sauce	0.020629
	engine	0.017747	group	0.006088	suite	0.022209	market	0.015368
	wheel	0.015192	genres	0.004815	time	0.020124	recipe	0.013451
	nissan	0.011769	pop	0.004508	motel	0.019339	pasta	0.009811
	recall	0.009794	musician	0.003245	hyatt	0.018888	jam	0.006312
	hybrid	0.007215	westlife	0.002751	plaza	0.010778	beef	0.004861

Table 4: Topic Evolution Discovered via WSSM(SPI)

	Period 1		**Period 2**		**Period 3**		**Period 4**		**Period 5**	
Vehicle	ford	0.032765	toyota	0.032086	car	0.034971	car	0.035798	car	0.035480
	recall	0.029876	wheel	0.029248	auto	0.034943	price	0.034654	price	0.034921
	silver	0.028352	parts	0.028533	price	0.029883	white	0.028841	white	0.028533
	toyota	0.027675	car	0.027579	sale	0.027591	brake	0.027393	toyota	0.027191
	jaguar	0.025987	bmw	0.025475	engine	0.024466	toyota	0.026344	brake	0.026614
	lexus	0.017211	engine	0.017141	brake	0.023131	engine	0.017113	engine	0.017747
	wheel	0.016098	price	0.016688	wheel	0.016134	wheel	0.015336	wheel	0.015192
	car	0.012766	lexus	0.012315	toyota	0.014871	recall	0.011930	nissan	0.011769
	price	0.010332	white	0.010415	white	0.011878	hybrid	0.009631	recall	0.009794
	hybrid	0.006008	hybrid	0.006815	hybrid	0.008673	nissan	0.007076	hybrid	0.007215

[4] Léon Bottou, *Online learning and stochastic approximations*, Online learning in neural networks (1998).

[5] Huanhuan Cao, Daxin Jiang, Jian Pei, Enhong Chen, and Hang Li, *Towards context-aware search by learning a very large variable length hidden markov model from search logs*, WWW, 2009.

[6] M.J. Carman, F. Crestani, M. Harvey, and M. Baillie, *Towards query log based personalization using topic models*, CIKM, 2010.

[7] Ludmila Cherkasova, *Scheduling strategy to improve response time for web applications*, High-Performance Computing and Networking, 1998.

[8] Graham Cormode and S Muthukrishnan, *Summarizing and mining skewed data streams*, SIAM, 2005, pp. 44–55.

[9] Muthukrishnan-S Cormode, Graham, *Approximating data with the count-min data structure*, IEEE Software (2012).

[10] Steve Fox, Kuldeep Karnawat, Mark Mydland, Susan Dumais, and Thomas White, *Evaluating implicit measures to improve web search*, TOIS (2005).

[11] Gregor Heinrich, *Parameter estimation for text analysis*, Web: http://www. arbylon. net/publications/text-est. pdf (2005).

[12] Matthew Hoffman, Francis R Bach, and David M Blei, *Online learning for latent dirichlet allocation*, NIPS, 2010.

[13] Liangjie Hong, Amr Ahmed, Siva Gurumurthy, Alexander J Smola, and Kostas Tsioutsiouliklis, *Discovering geographical topics in the twitter stream*, WWW, 2012.

[14] J. Huang and E. N. Efthimiadis, *Analyzing and evaluating query reformulation strategies in web search logs*, CIKM, 2009.

[15] Di Jiang, Jan Vosecky, Kenneth Wai-Ting Leung, and Wilfred Ng, *G-wstd: a framework for geographic web search topic discovery*, CIKM, 2012.

[16] Rosie Jones and Kristina Lisa Klinkner, *Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs*, 17th ACM conference on Information and knowledge management, 2008.

[17] D. Kang, D. Jiang, J. Pei, Z. Liao, X. Sun, and H. J. Choi, *Multidimensional mining of large-scale search logs: a topic-concept cube approach*, WSDM, 2011.

[18] Dongyeop Kang, Daxin Jiang, Jian Pei, Zhen Liao, Xiaohui Sun, and Ho-Jin Choi, *Multidimensional mining of large-scale search logs: a topic-concept cube approach*, WSDM, 2011.

[19] Jey Han Lau, Nigel Collier, and Timothy Baldwin, *On-line trend analysis with topic models:twitter trends detection topic model online.*, COLING, 2012.

[20] Kevin P Murphy, Yair Weiss, and Michael I Jordan, *Loopy belief propagation for approximate inference: An empirical study*, UAI, 1999.

[21] David Newman, Arthur Asuncion, Padhraic Smyth, and Max Welling, *Distributed algorithms for topic models*, The Journal of Machine Learning Research (2009).

[22] Thomas Niemann, *Sorting and searching algorithms: A cookbook*, Thomas Niemann, 2006.

[23] Patrick Pantel, Thomas Lin, and Michael Gamon, *Mining entity types from query logs via user intent modeling*, ACL, 2012.

[24] Ian Porteous, David Newman, Alexander Ihler, Arthur Asuncion, Padhraic Smyth, and Max Welling, *Fast collapsed gibbs sampling for latent dirichlet allocation*, SIGKDD, 2008.

[25] Daniel E Rose and Danny Levinson, *Understanding user goals in web search*, WWW, 2004.

[26] M. Rosen-Zvi, T. Griffiths, M. Steyvers, and P. Smyth, *The author-topic model for authors and documents*, UAI, 2004.

[27] Yelong Shen, Jun Yan, Shuicheng Yan, Lei Ji, Ning Liu, and Zheng Chen, *Sparse hidden-dynamics conditional random fields for user intent understanding*, WWW, 2011.

[28] S. Sizov, *Geofolk: latent spatial semantics in web 2.0 social media*, WSDM, 2010.

[29] Wei Song, Yu Zhang, Ting Liu, and Sheng Li, *Bridging topic modeling and personalized search*, COLING, 2010.

[30] Mark Steyvers and Tom Griffiths, *Probabilistic topic models*, Handbook of latent semantic analysis (2007).

[31] Yee W Teh, David Newman, and Max Welling, *A collapsed variational bayesian inference algorithm for latent dirichlet allocation*, NIPS, 2006.

[32] C. Wang, J. Wang, X. Xie, and W. Y. Ma, *Mining geographic knowledge using location aware topic model*, GIR, 2007.

[33] Kuansan Wang, Nikolas Gloy, and Xiaolong Li, *Inferring search behaviors using partially observable markov (pom) model*, WSDM, 2010.

[34] X. Wang and A. McCallum, *Topics over time: a non-markov continuous-time model of topical trends*, SIGKDD, 2006.

[35] Limin Yao, David Mimno, and Andrew McCallum, *Efficient methods for topic model inference on streaming document collections*, SIGKDD, 2009.

[36] Jia Zeng, W Cheung, and Jiming Liu, *Learning topic models by belief propagation*, PAMI (2011).

[37] Jia Zeng, Zhi-Qiang Liu, and Xiao-Qin Cao, *Online belief propagation for topic modeling*, arXiv (2012).

[38] Ke Zhai, Jordan Boyd-Graber, and Nima Asadi, *Using variational inference and mapreduce to scale topic modeling*, arXiv (2011).