TCS: Efficient Topic Discovery over Crowd-oriented Service Data

Yongxin Tong Caleb Chen Cao Lei Chen
Hong Kong University of Science & Technology, Hong Kong, China
{yxtong, caochen, leichen}@cse.ust.hk

ABSTRACT

In recent years, with the widespread usage of Web 2.0 techniques, crowdsourcing plays an important role in offering human intelligence in various service websites, such as Yahoo! Answer and Quora. With the increasing amount of crowd-oriented service data, an important task is to analyze latest hot topics and track topic evolution over time. However, the existing techniques in text mining cannot effectively work due to the unique structure of crowd-oriented service data, *task-response pairs*, which consists of the task and its corresponding responses. In particular, existing approaches become ineffective with the ever-increasing crowd-oriented service data that accumulate along the time. In this paper, we first study the problem of discovering topics over crowd-oriented service data. Then we propose a new probabilistic topic model, the *Topic Crowd Service Model* (TCS model), to effectively discover latent topics from massive crowd-oriented service data. In particular, in order to train TCS efficiently, we design a novel parameter inference algorithm, the *Bucket Parameter Estimation* (BPE), which utilizes belief propagation and a new sketching technique, called *Pairwise Sketch* (pSketch). Finally, we conduct extensive experiments to verify the effectiveness and efficiency of the TCS model and the BPE algorithm.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data Mining

General Terms

Design, Experimentation, Performance

Keywords

Crowd-oriented Service, Probabilistic Topic Model, Crowdsourcing

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
KDD'14, August 24–27, 2014, New York, NY, USA.
Copyright 2014 ACM 978-1-4503-2956-9/14/08 ...$15.00.
http://dx.doi.org/10.1145/2623330.2623647.

1. INTRODUCTION

Crowdsourcing refers to outsourcing works traditionally performed by an employee to an "undefined, generally large group of people in the form of an open call"[13]. In general, crowdsourcing-based service has a common framework: each employer (a.k.a the task publisher) poses a task, and then this task is responded or finished by many different and unknown crowd employees. Thus, the "task-response pairs" is the unique structure of crowdsourcing data. In this paper, we call information services provided by crowdsourcing, which include massive task-response pairs, as *crowd-oriented services*. Examples of crowd-oriented services include Yahoo! Answers, Quora, and Baidu Recommender, etc.

Given a crowd-oriented service system, one of the most significant problems is the "assignment problem"[11], namely how the crowd-oriented service system assigns a task to a right employee or helps an employee to find a right task. In order to enhance the accuracy of assignments, an effective method is to match them in terms of the semantic topic distribution of tasks and historical responses of employees. Thus, the problem of discovering topic is a fundamental task for crowd-oriented service systems. For example, stackoverflow[1] is a crowd-oriented website regarding programming techniques. By means of discovering the topic distributions of tasks and responses in stackoverflow, we can know which programming technique is the most popular one in current communities of programmers and guide the crowdsourcing platform to find hidden speciality of crowd employee according to their responses.

Although topic discovery is a basic operator for crowd-oriented service data, it is not trivial to capture hidden topics over crowd oriented service data using existing techniques of text mining. Back to the example about stackoverflow, Table 1 includes several real tasks and responses downloaded from the stackoverflow. The crowd-oriented service data in Table 1 contains infinite entries that are chronologically ordered and each entry contains a task and its corresponding responses (if any). We can observe that the first and second task-response entries discuss the topic related with Android-based developing techniques, and the third one is about iPhone techniques. If we use existing text mining techniques, such as Latent Dirichlet allocation (LDA) model, to discover the hidden topic distribution, a straightforward solution is to consider each task and response as a document and applies LDA model. For the first two entries in Table 1, this method can capture the topic about Android-based technique in the tasks and their responses since "An-

[1]http://stackoverflow.com/

Table 1: Crowd-oriented Service Data from the stackoverflow

ID	Tasks	Responses
1	Android application database to save images ...	Android SQLite database with multiple tables can...
2	How to use simpleadapter with activity Android...	An Android ListView with adapter is shown as follows...
3	Find mobilephones on hotspot networks in iPhone?...	iOS 7 system of Apple devices provide an IP address of ...
...	.....	.....

droid" occur in both of them. However, for the third entry, LDA model might not return similar topic distribution because there is no common words appeared in both of the task and the response.

Thus, for crowd-oriented service data, it is the necessary to discover hidden topics by considering both the task-response pair correlation and the specific semantic feature of each task and response. Furthermore, a real-time crowd-oriented service should be viewed as an online updateable data of crowd-oriented task-response pairs, which is referred as a series of consecutively submitted crowd-oriented tasks (task for short) and corresponding responses from crowds. Therefore, for these online service data, the efficiency of training the model is another major concern. The latent topics need to be discovered from massive crowd-oriented service data in real time. To achieve these goals in discovering topics over crowd-oriented service data, we have to address the following two challenges.

- Challenge 1: How to design an effective model to capture the task-response correlation over dynamic evolving crowd-oriented service data?
- Challenge 2: Facing the high volume of crowd-oriented service data, how to guarantee the training efficiency? There are two critical issues need to be addressed. The first problem is how to efficiently store and process massive task-response pairs. The other one is how to design an efficient parameter estimation approach, which can help deriving the proper parameter settings for the proposed topic detection model over crowd-oriented service data.

The aforementioned challenges reflect the high demand of an advanced probabilistic model which is tailored to meet the crowd data form and large data size. In answering this demand, we design and present the *Topic Crowd Service Model* (TCS), which features the capability of discovering latent topics from massive crowd-oriented service data with high accuracy. One essential niche of TCS is that it captures the the structure of task-response pair by preserving the mutual correlation. The massive size of incoming data, which is continuously increasing everyday, is the nightmare for most conventional parameter estimation methods such as collapsed Gibbs Sampling(GS)[20] and Variational Bayes(VB)[1], due to the high computational cost while traversing the dataset. It is observed that such computational cost attributes to the iterative behaviour of scanning the entire corpus and visit the complete topic space. This procedure leads to a linearly increasing computational time cost of the size of the data, the number of topics and the number of training iterations. In answering such challenge, we propose the *Block Parameter Estimation*(BPE), which is a fast parameter estimation method. The BPE features the *Pair Sketch*(pSketch) technique, which is a storage space reduction approach that alleviates the pain of updating new data trunks. Moreover, in order to achieve the convergence of parameter inference process, BPE integrates belief propagation[18] into the stochastic gradient descent framework [3], in which a series of online gradient updates lead to the stationary point of the likelihood function of TCS. Such design of the BPE brings in significantly speedup for the parameter inference process by simplifying the updates of TCS during the iterations: in each iteration, only a small portion of the crowd-oriented service data are selected as well as a part of topic space for message updating and passing. In sum, we make following major contributions:

- We design and present a new probabilistic topic model, the *Topic Crowd Service Model* (TCS model), which features the capability of discovering latent topics from massive crowd-oriented service data with high accuracy.
- We propose a fast parameter estimation algorithm, the *Bucket Parameter Estimation* (BPE), which utilizes a new sketching technique, called *pSketch*, and belief propagation to enhance the efficiency of the training process significantly.
- We verify the effectiveness and efficiency of the proposed methods through extensive experiments on real datasets. The experimental evaluation shows that TCS model and BPE algorithm outperform several existing probabilistic topic models and parameter estimation methods.

The rest of the paper is organized as follows. Our problem formulation is introduced in Section 2. In Section 3, we discuss the assumptions and generative process of the *Topic Crowd Service Model* (TCS). In addition, to train *TCS* efficiently, we present a new pairwise data-based sketch (pSkech) to quickly select the significant words in crowd-oriented service data in Section 4. Based on the significant words, we propose an effective parameter estimation algorithm, called *Bucket Parameter Estimation* (BPE), to train TCS in a specific bucket and multiple consecutive buckets of crowd-oriented service data in Section 5. We present the experimental results, related work and conclude the paper in Sections 6, 7, and 8, respectively.

2. PROBLEM FORMULATION

In this section, we formally define the related concepts and the problem of discovering hot topics over crowd-oriented service data. Let us begin with defining a few basic concepts as follows.

Definition 1 (Task-Response Pair). *Given a crowd-oriented task T_i, a set of corresponding responses $\{R_{i,1}, \cdots, R_{i,m}\}$, the arbitrary pair $(T_i, R_{i,j})$ for $j \in [1, m]$ is called a task-response pair.*

Table 2: Notation

Notation	Description
D	the set of all documents
W	the set of all words
W_T	the set of all words from tasks
W_R	the set of all words from responses
T	the set of distinct words from tasks
R	the set of distinct words from responses
K	the number of topics
d	document
s	sentence
z	topic
w	word
θ	multinomial distribution over topics
ϕ	multinomial distribution over words
α	Dirichlet prior vector for θ
β	Dirichlet prior vector for ϕ
$z^k_{d,s}$	the sentence s of document d is assigned to topic k
$z^k_{d,s,w}$	the word w in sentence s of document d is assigned to topic k
$n_{d,s,w}$	the number of w in sentence s of document d
λ_T	the significant threshold of words in tasks
λ_R	the significant threshold of words in responses
ρ_D	the proportion of documents for message passing
ρ_K	the proportion of topics for message passing
l_d	the related documents with the document d
ζ	the influence of related documents

DEFINITION 2 (CROWD-ORIENTED SERVICE DATA). *Let $CS=\{(T_1, R_{1,1}), \cdots, (T_n, R_{n,1}), \cdots, (T_n, R_{n,m})\}$ be a set of task-response pairs, where each task and response is a document. Each document d is represented by a subset of the collection of words $W=\{w_1, \cdots, w_{|W|}\}$. Given arbitrary task-response pair, a word-pair includes two words, one word is from the document of the task, the other word is from the document of the response.*

DEFINITION 3 (TOPIC). *A semantically coherent topic ϕ is a multinomial distribution of words $\{p(w|\phi)\}_{w \in W}$ with the constraint $\sum_{w \in W} p(w|\phi) = 1$.*

According to the definitions of related concepts, we can now formally define the major task in the problem of discovering topics over crowd-oriented service data as follows.

Discovering Topics over Crowd-oriented Service Data Problem: *Given the input of a crowd-oriented service data CS, we are required to infer the latent topics ϕ over in CS.*

Moreover, we list notations used in this paper in Table 2.

3. TOPIC CROWD SERVICE MODEL

In this section, we present a novel probabilistic model, *Topic Crowd Service Model* (TCS), for discovering topics over crowd-oriented service data. As shown in Table 1 and aforementioned definitions, each *document* is either a *task* or a *response from crowd employees*. Then, we illustrate the underlying logic of TCS. The generative process of TCS is shown in Algorithm 1.

Each document has a topic distribution. According to the pairs generated by pSketch, which will be introduced in Section 4, the topically coherency of each document is related. When composing the documents, the user first decides the topic that is aligned with his or her current task requirement and then selects some words according to the chosen topic. Notably, the topic distribution is determined by the document itself and the documents related by the corresponding task-response pairs. Furthermore, since each sentence is usually topically coherent, we impose the constraint that the words in each sentence should share the same topic, in order to capture semantic coherency.

Algorithm 1: Generative process of TCS

```
for each topic k ∈{1, ··· , K} do
    draw a word distribution φ_k ~ Dirichlet(β);
for each document d ∈{1, ··· , D} do
    draw topic distribution θ_d ~ Dirichlet(α);
    if d is a task then
        sample a response d' with regard to the number
        of sketch pairs between d and d';
    if d is a response then
        select the corresponding task d';
    generate new document topic distribution θ' by
    combining θ_d and θ_d';
    for each sentence s ∈ d do
        choose a topic z ~ Multinomial(θ');
        generate words w ~ Multinomial(φ_z);
```

Although existing parameter inference techniques, such as Gibbs sampling and variational Bayes, can be used to train the TCS model according to Algorithm 1, they have to spend higher computational cost. In order to enhance the efficiency of training process significantly, we will consider the problem of training the TCS model as a labelling problem. In other words, the training objective is equal to assign a set of topic labels to explain the observed data.

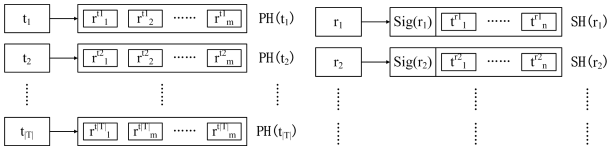
4. PAIRWISE SKETCH

As discussed in Section 3, the latent topic distribution is influenced by corresponding task-response pairs in the proposed TCS model. Thus, it is crucial for the training process to calculate frequencies of word pairs from task-response pairs. However, due to high-volume of online crowd-oriented service data, it is infeasible to count and store all word pairs due to the excessively high cost. Fortunately, word pairs with significant frequencies provide most information for discovering latent topics. So we propose to give priority to capture the contents of these pairs. Since finding significant word pairs is the foundation of our parameter estimation algorithm, we present a novel sketching structure, called *Pairwise Sketch* (*pSketch*), to efficiently select the significant word pairs from massive online task-response pairs. Before discussing the new data structure and algorithm, we first introduce several basic concepts and notations.

Given the set of all words of the given crowd-oriented service data W, it can be partitioned into two disjoint subsets, denoted by W_T and W_R, which consist of all words from tasks and responses, respectively. Moreover, the sets of distinct words in W_T and W_R are denoted by T and R, respectively. Then, a significant word pair is defined as follows.

DEFINITION 4 (SIGNIFICANT WORD PAIR). *Given a crowd-oriented service data CS, two significant thresholds λ_T and λ_R where $\lambda_T, \lambda_R \in (0, 1)$, a word pair (t,r) is a significant word pair if and only if $f(t,r) > \lceil \lambda_T f(t) \rceil$ and $f(t,r) > \lceil \lambda_R f(r) \rceil$, where f(t,r) means the frequency of the word pair (t,r), f(t) is the sum of frequencies of all word pairs with t as the word in tasks, and f(r) is similar.*

Please note that duplicates of a task in CS is only counted as one task. In general, $|W_R| >> |W_T|$, hence we set two dif-



(a) Primary Hashing (PH) (b) Secondary Hashing (SH)

Figure 1: Pairwise Sketch (pSketch)

ferent significant thresholds for them. In order to discover all significant word pairs for online crowd-oriented service data efficiently, a straightforward idea is to utilize existing streaming algorithms of finding frequent items, such as the CM-Sketch algorithm[7], the lossy-counting algorithm[15], the space-saving algorithm[17], and etc. However, the existing researches only handle single item rather than correlated pairs. Hence, we have to design a novel solution to find all significant word pairs. Inspired by the space-saving algorithm[17], which is one of the fastest streaming algorithms, we propose a novel sketching structure, called the *PairwiseSketch* (*pSketch*) and algorithm to find all significant word pairs. The pSketch is defined as follows.

Pairwise Sketch: It consists of two hashing structure components: the *primary hashing structure* (denoted PH) and the *secondary hashing structure* (denoted SH).

Primary Hashing Structure (PH): It includes |T| *primary hashing units* since the crowd-oriented service data has |T| distinct words from tasks. Each primary hashing unit (denoted PH(t_i)) uses t_i as the hashing key and maps a space-saving structure, which is also called the Stream-Summary in [17], to maintain the information about which words from responses are approximate significant for t_i. In other words, r_j is approximate significant for t_i if $f(t_i, r_j) > \lceil(\lambda_T - \epsilon)f(t)\rceil$. ϵ is the error ratio for λ_T. Moreover, based on [17], the number of elements of each space-saving structure is equal to $\lceil\frac{1}{\epsilon}\rceil$, namely $m = \lceil\frac{1}{\epsilon}\rceil$. Figure 1(a) shows a primary hashing structure.

Secondary Hashing Structure (SH): Similar to the *Primary Hashing Structure*, it consists of the set of *secondary hashing units*. Similarly, the number of elements of each space-saving structure, $n = \lceil\frac{1}{\delta}\rceil$ where δ is the error ratio for λ_R. However, there are two differences between the two hashing structures. On the one hand, the number of *secondary hashing units* is scalable rather than |R|. As discussed above, we choose the set of words from tasks as the primary hashing set since $|W_R| >> |W_T|$. Thus, our basic idea is to construct secondary hashing units only for the words from responses in current significant word pairs. On the other hand, an additional element, called the significant counter (denoted Sig(r_j)), appears in each secondary hashing units SH(r_j) and is used to filter out redundant secondary hashing units , where $Sig(r_j)$ records the number of currently significant word pairs from tasks with r_j. Once Sig(r_j)=0, r_j can be safely deleted from SH. Finally, in practice, in order to discover potential significant words from responses, we relax the monitoring requirement to the concept of ϵ-*significant words from responses*. Namely, a word r_j is ϵ-*significant* for a word t_i if $f(t_i, r_j) > \epsilon f(t_i)$. Figure 1(b) shows a secondary hashing structure.

According to the aforementioned pSketch structure, we design the significant word pair sketching algorithm in Algorithm 2. For each word pair (t,r), the algorithm first checks whether t constructs its primary hashing unit $PH(t)$ in lines 1-2. Then, the algorithm calls the space-saving algorithm subroutine to maintain frequencies of the word pairs

Algorithm 2: Significant Word Pairs Sketching

Input: a crowd-oriented service data CS, a error ratio ϵ for λ_T, a error ratio δ for λ_R,

```
1  for each word pair (t,r)∈CS do
2      if t does not construct its PH(t) in PH then
3          Construct PH(t);
4      Call Space-Saving(PH(t),r,ϵ);
5      if r becomes ϵ-significant for F(t) then
6          if SH(r) exists in SH then
7              Sig(r)← Sig(r)+1;
8      else
9          Construct SH(r) and Sig(r)← 1;
10     if r is not ϵ-significant for F(t) then
11         Sig(r)← Sig(r)-1;
12         if Sig(r)=0 then
13             Delete SH(r);
14     if SH(t)∈SH then
15         Call Space-Saving(SH(t),t,δ);
```

including t in line 4. If r becomes ϵ-significant for t and has $SH(r)$ in the current secondary hashing structure, the algorithm pluses one for Sig(r) in lines 5-7. Otherwise, $SH(r)$ is constructed in lines 8-9. When r is not ϵ-significant for t, Sig(r) is decreased by 1. In particular, the $SH(r)$ will be deleted from current secondary hashing structure if Sig(r) is equal to zero. In other words, the current r is never included by any significant word pairs. Finally, the algorithm also call the space-saving algorithm subroutine to maintain frequencies of the word pairs including r if $SH(r)$ is not deleted from SH. Hence, the pSketch structure not only maintains all significant word pairs but also returns them while traversing the sketch structure.

Computational Complexity Analysis: The time complexity and space complexity of Algorithm 2 are shown as follows. Since crowd-oriented service data is online in most cases, the number of word pairs may be currently unknown. Hence, we mainly analyze the time of processing each word pair in Algorithm 2. The processing time for each word pair in Algorithm 2 is an amortized constant time, which can be directly obtained because Algorithm 2 calls the space-saving subroutine in constant time, and the space-saving subroutine has the amortized constant processing time[17].

The space complexity of Algorithm 2 is $\mathcal{O}(\frac{1+\delta}{\epsilon\delta}|T|)$. In the following, the space usages of primary and secondary hash structures are analyzed, respectively. For primary hash structures, Algorithm 2 assigns a space-saving structure for each distinct word from task. Each space-saving structure spends $\mathcal{O}(\frac{1}{\epsilon})$ according to its definition[17]. Hence, the space usage of primary hash structures is totally $\mathcal{O}(\frac{|T|}{\epsilon})$. Furthermore, according to the definition of ϵ-significant word pairs, the maximum number of words that come from responses and satisfy $f(t,r) > \epsilon \cdot f(t)$ for any word t from tasks is $\frac{1}{\epsilon}$. Thus, the total space usage of secondary hash structures is $\frac{|T|}{\epsilon\delta}$. To sum up, the total space complexity of Algorithm 2 is $\mathcal{O}(\frac{|T|}{\epsilon} + \frac{|T|}{\epsilon\delta}) = \mathcal{O}(\frac{1+\delta}{\epsilon\delta}|T|)$.

5. PARAMETER ESTIMATION

In this section, we discuss the details of the *Bucket Parameter Estimation* (BPE). Section 5.1 first shows the procedure

of utilizing BPE to train TCS with crowd-oriented service data in a specific bucket. Section 5.2 also extends the BPE approach to the scenario of multiple consecutive buckets.

5.1 Parameter Estimation in Single Bucket

Section 4 introduces a novel sketching structure to select the significant task-response pairs in crowd-oriented service data. According to these task-response pairs, we propose a new latent parameter estimation approach in a bucket in this subsection, a. Since each sentence is the basic unit for topic assignment, we first aim to infer the probability that a sentence s of the document d is assigned to the topic k, namely the following formula,

$$P(z^k_{d,s}|\mathbf{z}^k_{d,-s}, \mathbf{n}_{d,-s}, \mathbf{z}^k_{\cdot,-s,w}, \mathbf{n}_{\cdot,-s,w}, \mathbf{l}_d) \tag{1}$$

where l_d denotes the related documents which include a word occurring with the other word of the current document d in at least a significant word pair in pSketch, $-s$ means all sentences in the current document d without s, and $\mathbf{z}_{d,-s}$ and $\mathbf{z}_{\cdot,-s,w}$ are all possible topic assignments of neighbouring variables.

Therefore, we denote the belief message by $\mu^k_{d,s}$, indicating a sentence s of the document d is generated by the topic k. It is represented by the following formula:

$$\mu^k_{d,s} \propto \frac{(1-\zeta)\mu^k_{d,-s} + \zeta\mu^k_{l_d} + \alpha_k}{\sum_{k'}\Big((1-\zeta)\mu^{k'}_{d,-s} + \zeta\mu^{k'}_{l_d} + \alpha_{k'}\Big)} \frac{\Gamma\Big(\sum_{r'}(n_{\cdot,-s,w'}\mu^k_{\cdot,-s,w'} + \beta_{w'})\Big)}{\Gamma\Big(\sum_{w'}(n_{\cdot,-s,w'}\mu^k_{\cdot,-s,w'} + \beta_{w'} + n_{d,s,w'})\Big)} \prod_{w\in s}\Big(\frac{\Gamma(n_{\cdot,-s,w}\mu^k_{\cdot,-s,w} + \beta_w + n_{d,s,w})}{\Gamma(n_{\cdot,-s,w}\mu^k_{\cdot,-s,w} + \beta_w)}\Big) \tag{2}$$

where ζ denotes a ratio to influence the intensity between the current document d and the related documents l_d. In addition, belief message $\mu^k_{d,s,w}$ that a word w in the sentence s can be also updated by $\mu^k_{d,s,w} = \mu^k_{d,s}$..

According to Equation 2, we need to perform a series of iterations to update beliefs and infer parameters. In order to guarantee the efficiency and effectiveness of training the TCS model, there are two crucial challenges: 1) What is the best strategy for the training process? 2) when should the iteration process terminates?

For the first challenge, our main idea is to choose the largest belief residual, which is denoted by $r^k_{d,s} = \Big|\mu^k_{d,s}(t) - \mu^k_{d,s}(t-1)\Big|$, as the updating goal in the updating process from the $(t-1)^{th}$ iteration to the t^{th} iteration. It is reasonable for the updating method in each iteration because the largest belief residuals speed up the convergence of iterations as much as possible. Based on the aforementioned updating strategy, we have to update all non-zero belief residuals, whose number is too huge. Hence, to further enhance the efficiency of the training process, we select only significant task-response pairs, topics, and documents in the training process according to the pSketch. Before introducing more details of our selection strategy, we first extend the concept of $r^k_{d,s}$ to the residuals of higher levels. We denote the residual of the document d on the topic k as $r^k_d = \sum_s r^k_{d,s}$. In other words, the residual of the document d on the topic k is equal to the sum of all residuals of sentences of the document d in the topic k. Similarly, we also denote the residual of the document d as $r_d = \sum_s r^k_d$. Then, we define two selected proportions, denoted by ρ_D and ρ_K, of documents and topics for message passing in each iteration.

Based on the above concepts, our selection updating strategy in each iteration is shown as follows.

- Step 1: We sort all r_d in a descending order and choosing $\rho_D \times D$ documents having the $\rho_D \times D$ largest r_d, where D is the number of documents.
- Step 2: For each selected document d, we sort all r^k_d in a descending order and choose $\rho_K \times K$ topics having the $\rho_K \times K$ largest r^k_d in d, where K is the number of topics.
- Step 3: We only update belief messages in the set of $\rho_D \times D$ documents and $\rho_K \times K$ topics.

According to the above selection updating strategy, we can obtain the normalized estimated messages as follows.

$$\hat{\mu}^k_{d,s}(t) = \frac{\mu^k_{d,s}(t)\sum_k \hat{\mu}^k_{d,s}(t-1)}{\sum_k \mu^k_{d,s}(t)} \tag{3}$$

$$\hat{\mu}^k_{d,s,w}(t) = \frac{\mu^k_{d,s,w}(t)\sum_k \hat{\mu}^k_{d,s,w}(t-1)}{\sum_k \mu^k_{d,s,w}(t)} \tag{4}$$

where $(t-1)$ and t are the previous iteration and the current iteration, respectively. Based on the aforementioned belief messages, we can infer the following parameters: the distribution of topics, θ, and the distribution of words, ϕ, respectively. Namely,

$$\theta^k_d = \frac{\mu^k_{d,\cdot} + \alpha_k}{\sum_{k'}\Big(\mu^{k'}_{d,\cdot} + \alpha_{k'}\Big)}. \tag{5}$$

$$\phi^k_w = \frac{\mu^k_{\cdot,\cdot,w} + \beta_w}{\sum_{w'}\Big(\mu^k_{\cdot,\cdot,w'} + \beta_{w'}\Big)}. \tag{6}$$

For the second challenge, the iteration process also plays a important role. We give two termination conditions for the iteration process. The first condition is a predefined number of iterations. The second condition is when the convergence of the iteration process occurs.

To sum up, based on the solutions to the above two challenges, Algorithm 3 presents the Bucket Parameter Estimation (BPE) algorithm. According to the pseudocode in Algorithm 3, BPE algorithm first initializes and normalizes $\mu^k_{d,s}(0)$ and $\mu^k_{d,s,w}(0)$ in line 2. The pSketch stores the frequencies of significant task-response pairs and assists the initialization of $\mu^k_{d,s,w}(0)$. Then, in each iteration, BPE algorithm sorts the documents according to corresponding r_d and selects $\rho_D \times D$ documents having the largest residuals in lines 3-10. For each selected documents, BPE algorithm also sorts topics and selects $\rho_K \times K$ topics having the largest residuals in lines 4-8. After the selection, this algorithm updates and normalizes $\mu^k_{d,s}(t)$ and $\mu^k_{d,s,w}(t)$ in line 5. In particular, please note that BPE algorithm has to sort and calculates all residuals of all documents and topics in the first iteration because there is no previous information. Finally, BPE algorithm terminates when one of two termination conditions, the maximum iteration number and the convergence of iteration, is satisfied.

5.2 Parameter Estimation in Multiple Consecutive Buckets.

Section 5.1 have introduced the Bucket Parameter Estimation algorithm within a bucket, we extend the BPE algorithm to the scenario of multiple consecutive buckets. In

Algorithm 3: Bucket Parameter Estimation (BPE)

for *each task-response pair in each document d and each topic k* **do**
  Random initialization and normalization $\mu^k_{d,s}(0)$ and $\mu^k_{d,s,w}(0)$ based on pSketch;
for *each iteration* **do**
  for *each document* **do**
    compute $\mu^k_{d,s}(t)$ and $\mu^k_{d,s,w}(t)$;
    compute $r^k_{d,s}(t)$, $r^k_d(t)$ and $r_d(t)$;
    Sort $r^k_d(t)$ in a descending order;
    Select the $\rho_K \times K$ documents having the largest residuals.
  Sort $r_d(t)$ in a descending order;
  Select the $\rho_D \times D$ documents having the largest residuals.

fact, there are many real applications for the scenario of multiple consecutive buckets. For example, incremental crowd-oriented service data, updating crowd-oriented service data, crowd-oriented service data steams, and so on. For such data, we can partition the complete data into multiple buckets, then extend BPE algorithm to infer latent topics and capture the evaluation of topics.

First of all, we define several new input parameters for the scenario of multiple consecutive buckets. We denote m as the index of multiple buckets and D_m as the number of documents in the m^{th} bucket. Please note that $m \in [1, \infty]$, $d \in [1, D_m]$, and $w \in [1, \infty]$. The indexes of multiple buckets and words reach infinity because infinity crowd-oriented service data are considered in the scenario of multiple consecutive buckets.

Then, we introduce how to extend BPE algorithm to the scenario of multiple consecutive buckets. Different from the updating strategy in the single bucket, the basic idea is that the updating parameters not only consider the information on the current bucket but also integrate the information on the previous buckets. Specifically, in the m^{th} bucket, the extended BPE algorithm first randomly initializes and normalizes the belief messages, denoted by $\mu^k_{d,s}[m]$, then initializes the sufficient statistics, denoted by $\Omega^k_w[m-1] = n_{\cdot,\cdot,w}[m-1]\mu^k_{\cdot,\cdot,w}[m-1]$. Hence, the belief message that a sentence s of the document d on the topic k in Equation 2 should be extended as follows,

$$\mu^k_{d,s}[m] \propto \frac{(1-\zeta)\mu^k_{d,-s}[m] + \zeta\mu^k_{l_d}[m] + \alpha_k}{\sum_{k'}\Big((1-\zeta)\mu^{k'}_{d,-s}[m] + \zeta\mu^{k'}_{l_d}[m] + \alpha_{k'}\Big)} \frac{\Gamma\Big(\sum_{w'}(n_{\cdot,-s,w'}[m]\mu^k_{\cdot,-s,w'}[m] + \Omega^k_{w'}[m-1] + \beta_{w'})\Big)}{\Gamma\Big(\sum_{w'}(n_{\cdot,-s,w'}[m]\mu^k_{\cdot,-s,w'}[m] + \Omega^k_{w'}[m-1] + \beta_{w'} + n_{d,s,w'})\Big)} \prod_{w\in s}\Big(\frac{\Gamma(n_{\cdot,-s,w}[m]\mu^k_{\cdot,-s,w}[m] + +\Omega^k_w[m-1] + \beta_w + n_{d,s,w})}{\Gamma(n_{\cdot,-s,w}[m]\mu^k_{\cdot,-s,w}[m] + \Omega^k_w[m-1] + \beta_w)}\Big) \tag{7}$$

According to the aforementioned belief message updating strategy, the BPE algorithm (Algorithm 3) can be extended to perform until at least one of the two termination conditions is satisfied.

6. EXPERIMENTAL STUDY

In this section, we evaluate the performance of TCS and BPE with a large-scale crowd-oriented service data. Section 6.1 describes the experimental setup. Section 6.2 presents the experimental results of BPE in terms of the efficiency. Section 6.3 measures the memory cost of BPE. Section 6.4 evaluates the effectiveness of TCS model with several standard metrics. Section 6.5 illustrates some results of topics and analyzes topic evolution in multiple consecutive buckets.

6.1 Experimental Setup

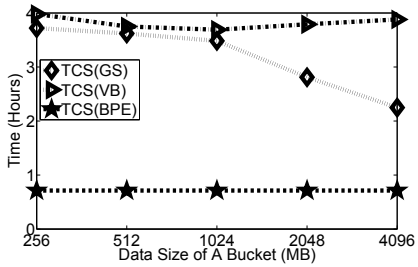
A crowd oriented service data (a.k.a question-answer data) from Yahoo is used as our experimental data. The data contains 142,612 questions and corresponding answers. Similar to [20], we set the hyperparameters as $\alpha = 2/K$ and $\beta = 0.01$. Furthermore, for sampling based parameter inference methods, we report the topic modeling results after 300 iterations, which practically ensures convergence in terms of perplexity that is a standard measure for evaluating the generalization of a probabilistic model [21].

6.2 Efficiency

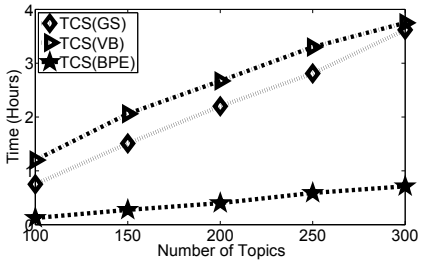
We first demonstrate the TCS performance concerning the efficiency aspect. Specifically, we compare BPE with a set of state-of-the-art practices including variational Bayes [1] (VB) and collapsed Gibbs sampling (GS) [10, 20]. We train all the models on the same dataset from pSketch in order to achieve a fair evaluation. Please refer to Figure 2 for the performance evaluation results. In Figure 2(a), the training time with the increase of the data size of a bucket is illustrated, where we set the topic amount $K = 300$. We summarize two findings via this experiment: 1) As shown in the figure, the training time of TCS(GS) slightly decreases with the data size of a bucket even it involves the additional cost of residual sorting, while that of TCS(VB) and TCS(BPE) increases. This is because the larger data size of a budget leads to a slightly faster convergence of the sampling based parameter inference methods. 2)TCS(BPE) shows less sensitivity to the change of the size of a bucket, which is a great character concerning online service with dynamic changes.

We then increase the topic amount K while fixing the data size of a bucket at 512MB, and we record the time cost of the three parameter inference methods. The results are shown in Figure 2(b), in which the running time of all three algorithms increase with K. However, since GS and VB require visiting all documents and the entire topic space, their time cost increase quickly when the data size grows larger. On the contrary, BPE only entails a subset of documents and a fraction of topic space, which qualifies itself as a fast topic modeling of massive crowd oriented service data.

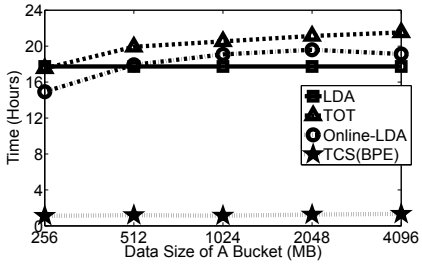
We then answer the question that TCS(BPE) outperforms the other topic models in terms of training efficiency. We vary the data size of a bucket and the topic amount, and then we evaluate the time consumption of different models and present the result in Figure 2(c) and 2(d). Here all baseline methods are trained on full data while TCS(BPE) is trained selectively. In Figure2(c) we set the topic amount to be 300 and the result demonstrates the superiority of TCS(BPE) over other models: 1) TCS is a relatively light-weight topic model and does not involve much complicated calculation; 2) TCS(BPE) utilizes pSketch to reduce the crowd oriented service data that need to be digested by the downstream topic modeling process; and 3) TCS(BPE) reduces the amount of documents and the scope of the topic space that need to be



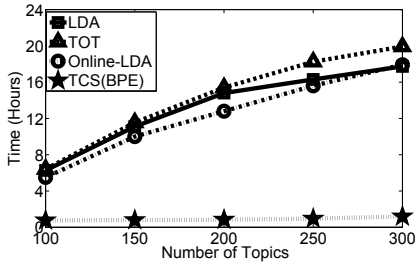
(a) Time vs Size (Different Parameter Inference Approaches)



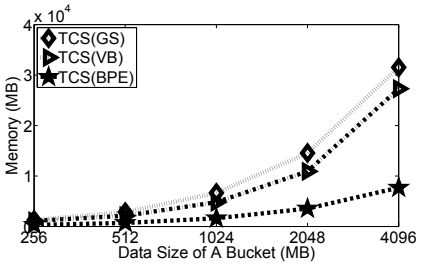
(b) Time vs # of Topics (Different Parameter Inference Approaches)



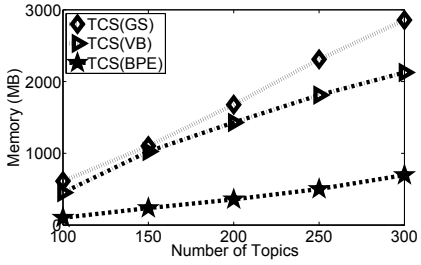
(c) Time vs Size (Different Topic Models)



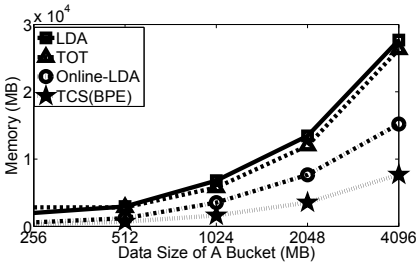
(d) Time vs # of Topics (Different Topic Models)



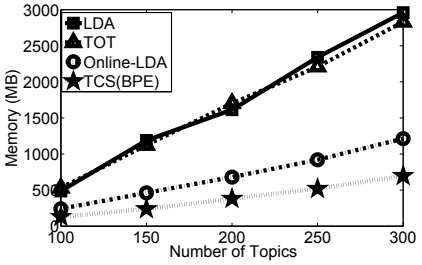
(e) Memory vs Size (Different Parameter Inference Approaches)



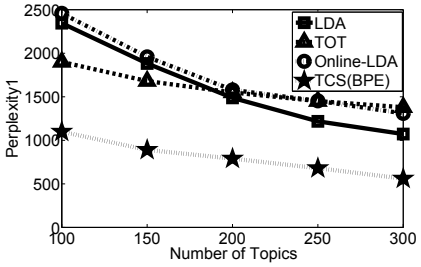
(f) Memory vs # of Topics (Different Parameter Inference Approaches)



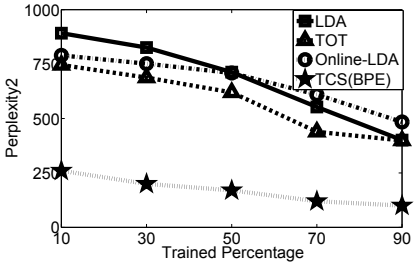
(g) Memory vs Size (Different Topic Models)



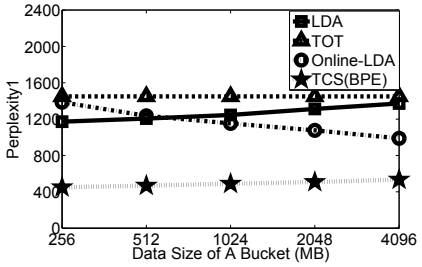
(h) Memory vs # of Topics (Different Topic Models)



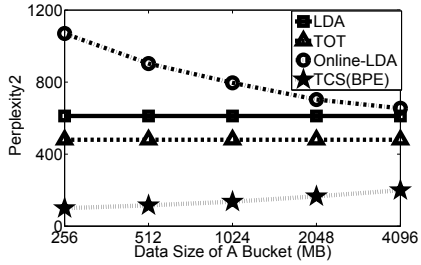
(i) Perplexity1 vs. # of Topics



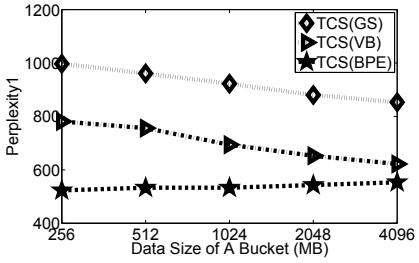
(j) Perplexity2 vs. Observed Percentage



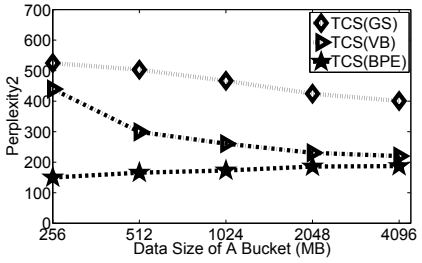
(k) Perplexity1 vs. Size (Different Topic Models)



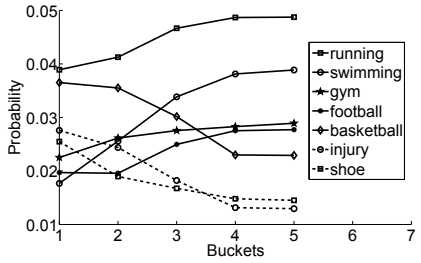
(l) Perplexity2 vs. Size (Different Topic Models)



(m) Perplexity1 vs. Size (Different Parameter Inference Approaches)



(n) Perplexity2 vs. Size (Different Parameter Inference Approaches)



(o) Topic Evaluation of TCS

Figure 2: Performance Evaluation

scanned in each iteration. We then fix the data size of a bucket to be 512MB while varying the topic amount K increasingly and evaluate the time cost. The results are shown in Figure 2(d), where the TCS(BPE) exhibits significantly better scalability over other methods with a rather moderate linear increase.

6.3 Memory Cost

Memory cost is another significant concern when evaluating topic modeling techniques. In this subsection, Figure 2(e) and Figure 2(f) show the results of comparing the memory cost of different parameter inference methods for training TCS. The BPE approach always outperforms both GS and VB in terms of memory cost.

Moreover, we also evaluate the performance of different topic models by varying the data size of a bucket. Please refer to Figure 2(g) as the results, where TCS(BPE) consumes much less memory resource than LDA and TOT. The reason is two-fold, firstly, classical topic models need to process all data and inevitably occupy large memory resources; secondly, small data size of each bucket helps other three topic models exploit the memory more effectively. For example, when the bucket size is set to 512MB, the typical memory cost of TCS(BPE) is only 617MB, which is much less than those consumed by the representative topic models. And this value even outperforms the result of Online-LDA, which demonstrates the merit of pSketch in Section 4. Then we demonstrate the memory cost in terms of a varying amount of topics while the data size of a bucket is set to be 512MB. The results are depicted in Figure 2(h), where linear increasing behavior is observed for every topic models. However, Online-LDA and TCS(BPE) shows lowest memory usage when the amount grows larger, which indicates its potential as a scalable practice.

6.4 Effectiveness

In this subsection, we discuss the effectiveness of TCS model. Specifically, the evaluation are conducted based on the concept of perplexity measurement[21]. Before we elaborate the details of the experiment, we first generalize its definition as below:

$$Perplexity1(\theta,\phi) = (\prod_{d=1}^{D}\prod_{i=1}^{N_d} p(w_i|\theta,\phi)^{\frac{-1}{\sum_{d=1}^{D}(N_d)}}, \quad (8)$$

$$Perplexity2(\theta,\phi) = (\prod_{d=1}^{D}\prod_{i=P+1}^{N_d} p(w_i|\theta,\phi,w_{a:P}))^{\frac{-1}{\sum_{d=1}^{D}(N_d-P)}}. \quad (9)$$

The difference between $Perplexity1$ and $Perplexity2$ is that the former one is used to describe the $held-out$ perplexity on the learned model ϕ, and the latter one is used to evaluate the effectiveness of *prediction* of the TCS model.

The design of TCS aims to achieve better generalization performance, which can be associated to a lower value of perplexity. In order to facilitate the empirical study, we integrate about ten thousand questions and corresponding responses into a dataset. We then evaluate the TCS capability of predicting unknown task and response words on such dataset, and we notice that the perplexity has a monotonically decreasing relationship with the likelihood of the dataset. Moreover, we adopt LDA [20] and TOT [25](both classical topic models) and Online-LDA [12](dynamic topic model) as the baseline approaches while following their corresponding parameter estimation methods. We summarize the results in Figure 2(i), where the lowest line(TCS) shows lower perplexity and therefore better capability to predicting unseen data comparing with the baselines. Note that for the online models(Online-LDA and TCS(BPE)), we assume the crowd oriented service data of each hour as a bucket.

Moreover, we aim to measure how effective the proposed models is in terms of predicting the future task and response words based on a portion of available tasks and responses. Specifically, given the task words $t_{1:P}$ from a user's log of tasks and response, we try to find out which model provides a better predictive distribution $p(t|t_{1:P})$ of the remaining words. In particular, eighty percent of tasks and responses data are set as the training data and the remaining twenty percent as the test data. The calculation of the perplexity is shown in Equation (9) and we summarize the comparison results in Figure 2(j). As shown in the figure, TCS(BPE) shows good capability to predict the future released tasks and responses given the historical tasks and responses.

We then present Figures 2(k) and (l) to show the perplexity1 and perplexity2 measurements while increasing the data size of a bucket. In Figure 2(k) the topic amount is fixed to 300. We can observe that all topic models is not sensitive for changing the data size of a bucket since they show stable value of the perplexity1 in terms of increasing data size of a bucket. Then, Figure 2(l) shows the perplexity2 measurement with the increase of the data size of a bucket. Not surprisingly, all the topic models remains stable on the perplexity2 measurement, and TCS(BPE) shows obvious superiority over other methods in terms of perplexity.

We further demonstrate in Figure 2(m) the perplexity1 measurement with increasingly varying data size of a bucket, while the topic amount K is set to 300. TCS(VB) exhibits lower perplexity when the data size of a bucket increases, because larger data size of bucket ushers in more robust online gradient descents for higher accuracy. On the contrary, TCS(GS) and TCS(BPE) often perform worse when the data size of a bucket increases, because smaller data size of a bucket helps correct the global biases. In all cases, TCS(BPE) achieves the lowest predictive perplexity, indicating the highest topic modeling accuracy. Figure 2(n) shows the performance of the perplexity2 measurement with the same setting above. We report similar result to that in Figure 2(m), where TCS(BPE) achieves highest topic modeling accuracy.

These experimental results above verify that TCS is a robust and effective topic model for crowd oriented service data in terms of the topic modeling accuracy.

6.5 Analysis of Topic Results and Evolution

Based on the topic modeling results of TCS, it is observed that TCS is designed to discover semantically meaningful topics by different parameter inference methods. The top ten words of four topics extracted by VB, GS and BPE on the same dataset are summarized in Table 3. All three parameter inference methods exhibit effectiveness in grouping semantically coherent task words together as topics, where their results observe high level overlapping of task words except slightly different word ranking. For instance, task words "running", "swimming" "gym", "football" and "basketball" are all contained in the topic *Sport*. And the rankings of these task words also show great resemblance with each other. Therefore we conclude that the discovered topics are comparable among all the three parameter inference algorithms, which means that using BPE to train TCS can

achieve paramount topic modeling accuracy while significantly better efficiency is observed.

The evolution of each topic is another informal but important measure of the success of topic models. We compare the topics that are discovered from consecutive different buckets and show the results in Figure 2(o),. Again we take the topic *Sport* as an example of topic evolution. In the first bucket, the task word "swimming"(solid line with circle marker) does not exist in the top five words. Then after receiving more data from the crowd oriented service data flow, the rank of "swimming" climbs from the sixth to the second in the following several buckets. The task words "gym" and "football" present similar trajectories where they gradually become more and more important in the topic of *Sport*. Word emergence and word perishment, in the meantime, is another important phenomenon in topic modeling results. For example, in the fourth bucket, the word "injury" appears in the topic for the first time and its rank reaches the 7th position in the fifth bucket. On the other hand, the word "shoe"(dotted lines) loses its importance to the topic as more and more crowd oriented service data are processed, and in the fourth bucket it eventually disappeared. All these results demonstrate the capability of TCS via BPE to detect the topic evolution.

7. RELATED WORK

In this section, we review the related work in two categories, crowd-oriented service computation and topic modeling problems.

7.1 Crowd-oriented Service Computation

Crowd-oriented service computation is a long-existing concept and has been practiced for centuries. With the emergence of Internet web service, especially the one that facilitates online question-and-answer websites like Yahoo! Answer and Baidu Recommender, crowd-oriented service starts to experience a new age where the source of human is broadened to a vast pool of crowds, instead of designated experts. This type of outsourcing to crowds, i.e. crowdsourcing, is now receiving countless success in many areas such as fund raising, logistics, monitoring and so on.

In crowd-oriented service applications, human cognitive abilities are mainly exploited in two types: voting among many options, and providing contents according to certain requirements. Most of basic queries in database [8] can be decomposed into simple voting as human tasks: such as filtering [5, 6, 19] into two-option voting (Yes or No), entity resolution [24], join [16], data cleaning[23], ranking [9], etc.

7.2 Topic Modeling

Since Blei et al. proposed the concept of topic modeling[2], topic modeling has attracted tremendous attention in both academic and industrial areas since its emergence. Especially along with the development of Web2.0 techniques, textual data plays a more and more important role in real-life application. Among these textual data, social network or social media like Facebook or Twitter exhibits great value due to their popularity and diversity in contents. Topic modeling is thus adopted to track emerging events in social communities [14] and to capture geographical topics [29].

Besides aforementioned wide applications, one of important issues is how to efficiently train the probabilistic models. The collapsed variational Bayesian inference for latent Dirichlet Allocation(LDA)[22] is proposed to beat its counterpart in terms of both computation cost and training accuracy. The work in [30, 31] then enables the classic loopy belief propagation for parameter estimation by considering the LDA as a factor graph. Topic distribution for new documents can also be inferred without retraining[28]. These parameter inference methods tackle the efficiency issue in training probabilistic topic models in different angles, but none of them are able to be easily adapted to meet the requirements from massive crowd oriented service data.

Moreover, another set of related researches with our work is the Community Question Answering (CQA). In this field, most previous work focused on bridging the lexical gap between the queried question and the historical questions [26]. Recently, a few work has studied how to utilize latent information to fill up the lexical gap [32, 4] and how to discover latent topics[27]. However, the biggest difference between researches of CQA and our work lies in the research objective. Our work focus on enhancing the efficiency of training process and save the space cost as much as possible through the proposed sketching techniques. However, the main goals of CQA are to bridge the lexical gap between the queried question and the historical questions and to recommend best answers to new questions.

To sum up, to the best of our knowledge, the TCS model together with the BPE algorithm and the pSketch structure is the first technique that systematically investigates the problem of topic discovery over massive crowd-oriented service data and provides solutions with solid performance.

8. CONCLUSIONS

In this paper, we study the problem of discovering latent topics over massive crowd-oriented service data efficiently. In order to guarantee the efficiency and effectiveness of the mining process, we design a novel probabilistic topic model, called *Topic Crowd Service Model*, which seamlessly incorporates a new data structure, called *Pairwise Sketch* (*pSketch*) and an efficient parameter estimation algorithm, called *Bucket Parameter Estimation* (*BPE*). We conduct extensive experiments in real data to verify the effectiveness and efficiency of TCS and BPE. In particular, we verify that BPE algorithm not only significantly enhances the efficiency of topic modeling but also decrease the memory cost than that of existing approaches.

9. ACKNOWLEDGMENTS

The authors thank the anonymous reviewers for their insightful and constructive comments. This work is supported in part by the Hong Kong RGC Project N_HKUST637/13, National Grand Fundamental Research 973 Program of China under Grant 2012-CB316200, National Natural Science Foundation of China (NSFC) Grant No. 61232018, Microsoft Research Asia Gift Grant, Microsoft Research Asia Fellowship 2012 and Google Faculty Award 2013.

10. REFERENCES

[1] H. Attias. Inferring parameters and structure of latent variable models by variational bayes. In *UAI*, 1999.

[2] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[3] L. Bottou. Online learning and stochastic approximations. *Online learning in neural networks*, 1998.

[4] L. Cai, G. Zhou, K. Liu, and J. Zhao. Learning the latent topics for question retrieval in community qa. In *IJCNLP*, pages 273–281, 2011.

Table 3: Evaluation of the Topic Results

Topics	Bucket Parameter Estimation (BPE)		Gibbs Sampling (GS)		Varitional Bayes	
Computer Programming	language	0.049678	language	0.045369	language	0.041782
	software	0.040128	code	0.042315	code	0.039274
	Java	0.038941	C++	0.035129	Java	0.036855
	code	0.036762	Java	0.031524	software	0.033937
	C++	0.032754	python	0.030671	python	0.031726
Restaurant	menu	0.051437	menu	0.059744	menu	0.058472
	food	0.045139	food	0.050024	food	0.049647
	flavour	0.039116	drink	0.042367	drink	0.046438
	drink	0.035406	pizza	0.036694	flavour	0.041082
	pizza	0.032773	wine	0.030431	pizza	0.037568
Sport	running	0.048745	running	0.031074	running	0.058742
	swimming	0.038868	swimming	0.020345	swimming	0.051286
	gym	0.028912	football	0.019074	basketball	0.047921
	football	0.027712	basketball	0.016109	football	0.041235
	basketball	0.022902	gym	0.014321	gym	0.030039
Tourism	visitor	0.068868	visitor	0.060202	visitor	0.061964
	hotel	0.053937	hotel	0.052887	hotel	0.050129
	beach	0.048002	price	0.047609	price	0.042129
	price	0.038868	museum	0.041298	beach	0.036812
	shop	0.034149	beach	0.031011	museum	0.029432
Weather	hot	0.039623	hot	0.041102	hot	0.038729
	cold	0.036009	stormy	0.037210	cold	0.032808
	stormy	0.030854	cold	0.031852	rain	0.029458
	earth	0.028791	wind	0.026197	stormy	0.024912
	wind	0.026486	rain	0.023843	wind	0.019247

[5] C. C. Cao, J. She, Y. Tong, and L. Chen. Whom to ask? jury selection for decision making tasks on micro-blog services. *PVLDB*, 5(11):1495–1506, 2012.

[6] C. C. Cao, Y. Tong, L. Chen, and H. V. Jagadish. Wisemarket: a new paradigm for managing wisdom of online social users. In *KDD*, pages 455–463, 2013.

[7] G. Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *J. Algorithms*, 55(1):58–75, 2005.

[8] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin. Crowddb: answering queries with crowdsourcing. In *SIGMOD*, pages 61–72, 2011.

[9] S. Guo, A. G. Parameswaran, and H. Garcia-Molina. So who won?: dynamic max discovery with the crowd. In *SIGMOD*, pages 385–396, 2012.

[10] G. Heinrich. Parameter estimation for text analysis. *Web: http://www. arbylon. net/publications/text-est. pdf*, 2005.

[11] C.-J. Ho, S. Jabbari, and J. W. Vaughan. Adaptive task assignment for crowdsourced classification. In *ICML (1)*, pages 534–542, 2013.

[12] M. Hoffman, F. R. Bach, and D. M. Blei. Online learning for latent dirichlet allocation. In *NIPS*, 2010.

[13] J. Howe. *Crowdsourcing: Why the Power of the Crowd Is Driving the Future of Business*. Crown Business, 2009.

[14] C. X. Lin, B. Zhao, Q. Mei, and J. Han. Pet: a statistical model for popular events tracking in social communities. In *KDD*, pages 929–938, 2010.

[15] G. S. Manku and R. Motwani. Approximate frequency counts over data streams. In *VLDB*, pages 346–357, 2002.

[16] A. Marcus, E. Wu, D. R. Karger, S. Madden, and R. C. Miller. Human-powered sorts and joins. *PVLDB*, 5(1):13–24, 2011.

[17] A. Metwally, D. Agrawal, and A. El Abbadi. An integrated efficient solution for computing frequent and top-k elements in data streams. *ACM Trans. Database Syst.*, 31(3):1095–1133, 2006.

[18] K. P. Murphy, Y. Weiss, and M. I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *UAI*, 1999.

[19] A. G. Parameswaran, H. Garcia-Molina, H. Park, N. Polyzotis, A. Ramesh, and J. Widom. Crowdscreen: algorithms for filtering data with humans. In *SIGMOD*, pages 361–372, 2012.

[20] I. Porteous, D. Newman, A. Ihler, A. Asuncion, P. Smyth, and M. Welling. Fast collapsed gibbs sampling for latent dirichlet allocation. In *SIGKDD*, 2008.

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

[22] Y. W. Teh, D. Newman, and M. Welling. A collapsed variational bayesian inference algorithm for latent dirichlet allocation. In *NIPS*, 2006.

[23] Y. Tong, C. C. Cao, C. J. Zhang, Y. Li, and L. Chen. Crowdcleaner: Data cleaning for multi-version data on the web via crowdsourcing. In *ICDE*, pages 1182–1185, 2014.

[24] J. Wang, T. Kraska, M. J. Franklin, and J. Feng. Crowder: Crowdsourcing entity resolution. *PVLDB*, 5(11):1483–1494, 2012.

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

[26] X. Xue, J. Jeon, and W. B. Croft. Retrieval models for question and answer archives. In *SIGIR*, pages 475–482, 2008.

[27] L. Yang, M. Qiu, S. Gottipati, F. Zhu, J. Jiang, H. Sun, and Z. Chen. Cqarank: jointly model topics and expertise in community question answering. In *CIKM*, pages 99–108, 2013.

[28] L. Yao, D. Mimno, and A. McCallum. Efficient methods for topic model inference on streaming document collections. In *SIGKDD*, 2009.

[29] Z. Yin, L. Cao, J. Han, C. Zhai, and T. S. Huang. Geographical topic discovery and comparison. In *WWW*, pages 247–256, 2011.

[30] J. Zeng, W. Cheung, and J. Liu. Learning topic models by belief propagation. *PAMI*, 2011.

[31] J. Zeng, Z.-Q. Liu, and X.-Q. Cao. Online belief propagation for topic modeling. *arXiv*, 2012.

[32] T. C. Zhou, C.-Y. Lin, I. King, M. R. Lyu, Y.-I. Song, and Y. Cao. Learning to suggest questions in online forums. In *AAAI*, 2011.