An Efficient Method for High Quality and Cohesive Topical Phrase Mining

Bing Li, Xiaochun Yang*, Rui Zhou, Bin Wang, Chengfei Liu, Yanchun Zhang

Abstract—A phrase is a natural, meaningful, and essential semantic unit. In topic modeling, visualizing phrases for individual topics is an effective way to explore and understand unstructured text corpora. However, from phrase quality and topical cohesion perspectives, the outcomes of existing approaches remain to be improved. Usually, the process of topical phrase mining is twofold: phrase mining and topic modeling. For phrase mining, existing approaches often suffer from order sensitive and inappropriate segmentation problems, which make them often extract inferior quality phrases. For topic modeling, traditional topic models do not fully consider the constraints induced by phrases, which may weaken the cohesion. Moreover, existing approaches often suffer from losing domain terminologies since they neglect the impact of domain-level topical distribution. In this paper, we propose an efficient method for high quality and cohesive topical phrase mining. A high quality phrase should satisfy frequency, phraseness, completeness, and appropriateness criteria. In our framework, we integrate quality guaranteed phrase mining method, a novel topic model incorporating the constraint of phrases, and a novel document clustering method into an iterative framework to improve both phrase quality and topical cohesion. We also describe efficient algorithmic designs to execute these methods efficiently. The empirical verification demonstrates that our method outperforms the state-of-the-art methods from the aspects of both interpretability and efficiency.

Index Terms—Topical Phrase Mining, Phrase Mining, Chunking, Topic Model.

1 Introduction

Topical phrase mining refers to automatically extracting phrases which grouped by individual themes from given text corpora. It is of high value to enhance the power and efficiency to facilitate human to explore and understand a large amount of unstructured text data. One example is that if researchers could find phrases among a research field appearing with high frequencies in related proceedings in different years, they will be able to have an insight into the academic trend of that research field. Topical phrase mining is not only an important step in established fields of information retrieval and text analytics, but also is critical in various tasks in emerging applications, including topic detection and tracking [1], social event discovery [2], news recommendation system, and document summarization [3].

Usually, the process of topical phrase mining is twofold: phrase mining and topic modeling. These two stages not only directly affect the quality of discovered phrases and the cohesion of topics, but also, they may interact and indirectly impact each other's outcomes, e.g., low quality phrases (incomplete or meaningless) may cause misleading topical assignment in topic modeling. However, from phrase quality and topical cohesion perspectives, the outcomes of existing approaches remain to be improved.

From phrase quality perspective, existing phrase mining methods [4–11] often produce low quality phrases. A high quality phrase should satisfy frequency, phrasegrness, completeness, and appropriateness criteria. Phrase mining is originated from the natural language processing (NLP) community, which utilizes predefined linguistic rules that rely on part-of-speech (POS) tagging or parsing trees [4, 5] to generate phrases. Such NLP based methods are commonly language-dependent and need texts to comply with grammar-rules, so it is not easy for them to be migrated to other languages and not suitable for analyzing some newly emerging and grammar-free text data, such as twitters, academic papers and query logs. In the hope to overcome the disadvantages of NLP based methods, there are many data-driven approaches that have been proposed in this area. Data-driven methods primarily view phrase mining as a frequent pattern mining problem [6, 7]. A phrase is extracted if it is constituted by the longest word sequence whose frequency is larger than a given threshold. Inevitably, extracting word sequence according to frequency is prone to produce many false phrases. Recently, researchers have sought for a kind of general, yet powerful phrase mining method. A variety of statistic-based methods [8–10] have been proposed to improve phrases quality by ranking candidate phrases. A more recent work [11] considers integrating phrasal segmentation with phrase quality estimation to estimate rectified phrase frequency to further improve phrase quality.

However, due to suffering from order sensitive and inappropriate segmentation, the outcome of existing methods is still inadequate. Below we use Table 1 to show the deficiencies of the existing methods by using significance scores Sig_score extracted from a corpus, 5Conf. [1] We compared two phrases using different processing orders based on 5Conf. Data in Table 1 is derived from the result of an existing method [9] which heuristically merges words under t-test score (i.e., a statistical hypothesis test to measure whether its actual occurrence significantly different from expected occurrence). The expected occurrence of phrase $Pr = w1 \oplus w2$ is calculated by $\frac{f(w_1)\times f(w_2)}{N}$, where $f(w_1)$ and $f(w_2)$ are word frequencies of w_1 and w_2 in the corpus, respectively, and N is the total number of words in the corpus. The method [9] allows users to specify a threshold of a significance score Sig_score(Pr) of a phrase Pr, which is the statistical significance of taking a group of

(*)-Corresponding author.

- *Bing Li, Xiaochun Yang, and Bin Wang are with the School of Computer Science and Engineering, Northeastern University, Shenyang, Liaoning 110819, China. E-mail: libing@stumail.neu.edu.cn, {yangxc,binwang}@mail.neu.edu.cn.*
- *Rui Zhou and Chengfei Liu are with Department of Computer Science and Software Engineering, Swinburne University of Technology, Melbourne VIC3122, Australia. E-mail: {rzhou,cliu}@swin.edu.au*
- *Yanchun Zhang is with the Centre for Applied Informatics, College of Engineering and Science, Victoria University, Melbourne VIC3011, Australia. E-mail: Yanchun.Zhang@vu.edu.au*

Manuscript received XXX XX, XXXX; revised XXX XX, XXXX.

1. http://elkishk2.web.engr.illinois.edu

words as a phrase. It is measured by comparing the actual frequency with the expected occurrence. A larger value of Sig_score(Pr) indicates the word sequence Pr has higher possibility to be a whole unit (phrase) than other sequences, and vice versa.

TABLE 1
Examples of heuristic methods run on a corpus, 5Conf.

Sig_score \ Token / Order	①Gaussian	②Mixture	③Model	Phrase?
①②:③	6391.62		15.75*	No
②③:①	2257.84*	23.96		Yes
	①Peer to	②Peer	③Data	
①②:③	186927.32		10.06*	No
②③:①	7034.07*	48.71		Yes

Here we set a threshold Sig_score $= 16$.

(1) *Order sensitive*. Assume `Gaussian Mixture Model` is a high quality phrase since it is complete in semantic. By choosing the merge order ①②:③, as shown in Table 1, existing approaches heuristically merge `Gaussian` and `Mixture` firstly, since the order shows a higher t-test score 6391.62 to achieve a local optimum comparing with the score 23.96 by using the order ②③:①. However, if the threshold Sig_score $= 16$, the complete phrase `Gaussian Mixture Model` failed to be extracted by using the order ①②:③ since the final core 15.75 is less than the given threshold 16 (we use symbol $*$ to denote the score of the whole phrase under the given merge order). Instead, the merge order ②③:① could have this phrase extracted. For the second phrase `Peer to Peer Data`, by using the same corpus, we got the same conclusion. Consequently, the completeness of extracted phrases highly depends on the merging order of the merging heuristics. The incompleteness brought by traditional approaches will cause incomplete semantics and may produce very general phrases. For instance, phrase `Mixture Model` has many explanations, such as Gaussian Mixture Model, Finite Mixture Model, or Interactive Mixture Model, whereas by phrase `Gaussian Mixture Model`, one explicitly refers to the very probabilistic model.

(2) *Inappropriate segmentation*. For the word sequence `Gaussian Mixture Model Selection`, it contains two quality phrases `Gaussian Mixture Model` and `Model Selection` since they both have high statistic scores. However, these two quality phrases are overlapping in the sequence. In the scenario of text chunking, the word `model` can only belong to one of these two phrases, i.e., s_1 = `Gaussian Mixture Model | Selection` or s_2 = `Gaussian Mixture | Model Selection`. Existing approaches which only consider intra-coocurrence (e.g., phrase frequency and phrase length) prefer to choose sequence s_2 , since both `Gaussion Mixture` and `Model Selection` have high frequencies. However, `Gaussian Mixture Model` should be the right choice for it is a whole function unit as an adjective, while `Gaussian Mixture` is obviously an incomplete phase.

From topical cohesion perspective, traditional topic models, such as LDA, assume words are generated independently from each other, i.e. "bag-of-words" assumption. Under this assumption, a phrase is regarded as an independent "word", which may lead to the loss of its specific meaning, and as a result, the impact of phrases is ignored.

To address the topic assignment problem associated with phrase, some existing methods such as PhraseLDA [9] uses an undirected clique to model the stronger correlation of words in the same phrase on top of the "bag-of-phrases" assumption. To be specific, words in the same phrase form a clique, and PhraseLDA imposes the same latent topic on the words in the same clique. However, it is not enough to consider only the correlation of a phrase and its words. A phrase as a whole may carry lexical meaning that is beyond the sum of its individual words. For example, the phrase `max pooling` has a meaning beyong the word "max" or "pooling". Thus, it would be inappropriate to enforce words in the same phrase to inherit the same topic like PhraseLDA does, since long noun phrases sometimes do have components indicative of different topics [12].

Moreover, existing approaches neglect a fact that some phrases are only valid in certain domains. Usually, the texts within a corpus often come from more than one domain, and each domain may contain its own terminologies. These domain-specific terminologies may only appear frequently within certain domains but not in others, making them less possible to be extracted in the entire corpus where their occurrence frequency is diluted by the other domains, as Table 2 demonstrates.

TABLE 2
An example of phrase significance score (t-statistic) on machine learning, database, math, data mining domains that derived from 5Conf dataset.

Phrase	ML	DB	MA	DM	ALL
support vector machine	89361	75.34	0	34.19	47766
eigen vector	74.52	0	3544	0	11.65
bit vector	0	398.44	0.67	0	5.244
social networks	42.43	7.36	0	753	76.205

In Table 2, the phrases `support vector machine`, `eigen vector`, `bit vector`, and `social networks` are estimated to belong to machine learning (ML) , math (MA), database (DB), and data mining (DM) domains, respectively. Even though some phrases (e.g., `support vector machine` and `social networks`) can achieve a high enough significance in the entire corpus, while others such as `bit vector` and `eigen vector` cannot. Consequently, it is hard for them to be mined as phrases in the entire corpus, albeit actually they both are common terminologies in their own domains.

Besides effectiveness, efficiency is also very important to topical phrase mining, especially for the applications that need timely analysis, such as topic-tracking [1], social event discovery [2], and news recommendation system. Take Twitter as an example, the volume of tweets grew at increasingly high rates from its launch in 2006 to 2010, approaching around $1,000\%$ gain in yearly volume[2]. Currently, over $350,000$ tweets are generated on Twitter per minute. Unfortunately, most existing approaches [11–14] often suffer from low efficiency as they cannot support such high throughput tasks.

In order to effectively and efficiently mine topical phrases and improve phrase quality and topical cohesion, we propose a *Cohesive and Quality Topical Phrase Mining* (CQMine) framework, which automatically clusters documents with a more sensible topic model, and improves the quality of phrases by adopting more accurate and rigorous mining approaches. Moreover, our quality phrase mining approach can be solely used to mine phrases.

The main contributions of this paper are as follows:

- We propose effective and efficient quality phrase mining approaches. By eliminating order sensitive and avoiding inappropriate segmentation, our approaches could guarantee the quality of extracted phrases. Moreover, we also design effective algorithms to accelerate the processing.
- We propose a novel topic model to address topic assignment problem associated with idiomatic phrases to improve the cohesion of topical phrases.
- Considering the fact that some phrases are only valid in certain domains, we propose an iterative framework to facilitate more accurate domain terminologies finding.
- Experimental evaluation and case study demonstrate that our method is of high interpretability and efficiency compared with the state-of-the-art methods.

2. http://www.internetlivestats.com/twitter-statistics/

The rest of the paper is organized as follows: Section 2 introduces the preliminaries and the problem definition. Section 3 outlines our CQPTM frameworks. Section 4 discusses the quality phrase mining. Section 5 introduces our topic model. Section 6 discusses the document clustering. Section 7 reports the result of empirical verification. Section 8 provides a detailed survey of related works. Section 9 concludes the paper.

2 Preliminaries and Problem Definition

This section introduces some definitions and notations.

Definition 1. *A document d is a sequence of words: $d = w_1, w_2, ..., w_n$, where n is the document length.*

Definition 2. *A phrase Pr can be formally represented as a consecutive sequence of words from j-th position in a document d: $Pr = d[j, j+l-1]$, where $d[j, j+l-1] = w_j, w_{j+1}, ..., w_{j+l-1}$, and l is the phrase length.*

In this paper, we mainly focus on consecutive-word phrases. It is also the core issue in phrase mining and the main research problem of some recent works [9, 11, 12, 15].

A "quality" phrase should be natural and meaningful. There is no universally accepted definition of phrase quality. In this paper, we define a quality phrase as a phrase satisfying the following four criteria.

(i) Frequency [9]: This criterion is based on the observation that an infrequent phrase is likely to be not important. For example, phrases such as `Social Networks` and `Support Vector Machine` tend to be "good" phrases since they are frequently used. Also, in a statistical perspective, high frequency is important to satisfy the theoretical minimal sample size.

Definition 3. *Frequency. If a phrase Pr_i is a frequent phrase, it must have $f(Pr_i) > ft$, where $f(Pr_i)$ denotes Pr_i's frequency and ft is a certain frequency threshold.*

(ii) Phraseness [13]: If adjacent phrases co-occur more significantly than expected under a given statistical significant level, these phrases should be merged into a longer phrase. For example, `Knowledge` and `Bases` can be merged into a longer phrase `Knowledge Bases` because they co-occur more frequently than expected.

Definition 4. *Phraseness. Phrase $Pr(|Pr| > 1)$ satisfies phraseness requirement, if and only if $\exists Pr_i, Pr_j$, such that $Pr = Pr_i \oplus Pr_j$ and $\upsilon(Pr_i, Pr_j) = true$, where υ is a boolean function to decide whether Pr_i, Pr_j could be merged into Pr under a certain statistical significant measurement, e.g., χ^2 test, z-test, or mutual information.*

(iii) Completeness: In this paper, we follow the definition of completeness criterion in [11], which considers completeness of a phrase in a specific text. To be specific, a longer phrase can express a certain semantics more precisely if it also satisfies Frequency and Phraseness criteria. For example, in text "... a Gaussian mixture model is a probabilistic model that assumes ..." `Gaussian mixture model` is identified as complete phrase rather than `Gaussian mixture` or `mixture model`, whereas in another text "... in statistics, a mixture model is a ...", `mixture model` is also complete since there is no phrase that could cover it as a part in this text. That is, we do not allow a proper subsequence of a phrase to be a phrase in a specific text.

Definition 5. *Completeness. For a specific document d, assume $d[i, i+l] = w_i, w_{i+1}, ..., w_{i+l}$ could form a phrase Pr_i under Frequency and Phraseness criteria. Pr_i satisfies completeness criterion if and only if $\nexists Pr_j = d[j, j+k] = w_j, w_{j+1}, ..., w_{j+k}$, such that $[i, i+l]$ is a sub-range of $[j, j+k]$.*

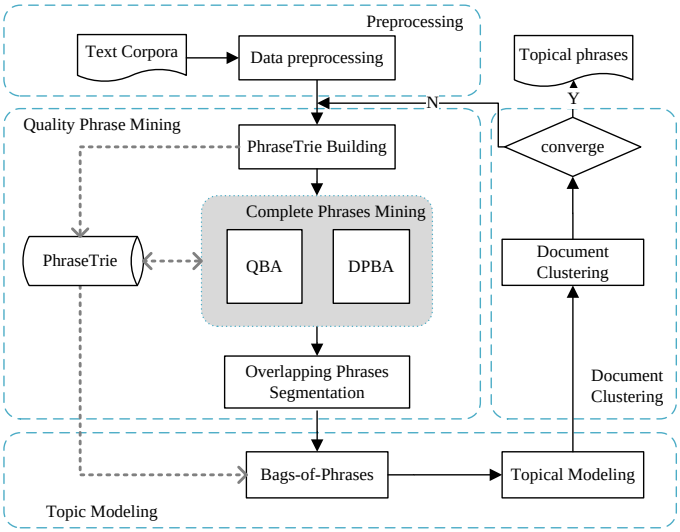


Fig. 1. CQMine framework

(iv) Appropriateness: If complete phrases are overlapping in a specific text, an appropriate segmentation should ensure the segmented phrases are disjoint and each of them is semantically independent.

Definition 6. *Appropriateness. Let $S = w_1, w_2, ..., w_{|S|}$ be the sequence which embodied a set of overlapping phrases, and its overlapping range is $\mathbb{L}$ ($\mathbb{L} \subseteq [1, |S|]$). An appropriate overlapping phrases segmentation (OPS) aims to find a set of split positions $P = \{b_1, b_2, ..., b_m\}$ ($b_1 = 1$, $b_m = |S|$, and $b_i \in \mathbb{L}$ $(i \notin \{1, m\})$) to split S into $m-1$ disjoint subsequences $s_1, s_2, ..., s_{m-1}$, and each subsequence $s_1 = w_{b_i}, ...w_{b_{i+1}-1}$ is a sequence of words. The goal of OPS is to maximize the following joint probability:*

$$p(P^*, S) = \prod_{i=1}^{m-1} p(s_i, b_{i+1}|b_i) \times p({b_i}^*|s_i, s_{i-1}), \qquad (1)$$

where $p(s_i, b_{i+1}|b_i)$ denotes the conditional probability of observing the sequence s_i given the i-th split position b_i, it reflects the intra-coocurrence of phrases. $p(b_i^*|s_i, s_{i-1})$ is an isolation indicator of the i-th split position b_i given the previous sequence s_{i-1} and the next sequence s_i. Note that, we have $p(b_1^*|\cdot) = 1$.

The details of how Eq. (1) is concluded will be discussed in Section 4.3.

Follow the definition in LDA, we define a topic as multinomial distribution over a fixed vocabulary, and on top of this, we define topical document cluster as follows:

Definition 7. *A topical document cluster C_k is defines as a collection of documents $C_k = \{d_0^k, ..., d_{s^k}^k\}$ in which all documents have similar document-topic distribution.*

If two documents share similar topic distributions, they should belong to the same cluster with a high probability. We will discuss how to measure this similarity in Section 6.

3 CQMine Framework

The framework of CQMine is shown in Fig. 1, the whole process consists of four major stages: preprocessing, quality phrase mining, topic modeling, and document clustering. The preprocessing stage includes some trivial preprocessing steps such as tokenization, dropping stop-words and stemming, which can be readily implemented by using existing tools [16, 17], and therefore we will not discuss in this paper.

The quality phrase mining stage contains three steps: Firstly, a *PhraseTrie* is built to count all possible phrases' frequencies.Then, a complete phrase mining algorithm (DPBA or QBA, they will be discussed in Section 4) is applied to mine complete phrases, which will be under the guidance of a statistics-based measurement to satisfy phraseness criterion. During phrase mining, the mined phrases are stored in *PhraseTrie* to avoid recomputing duplicate phrases. Finally, to guarantee the appropriateness requirement, for each document, CQMine needs to check if it contains overlapping phrases, if so, we will partition them into non-overlapping phrases by utilizing an effective and efficient overlapping phrases segmentation algorithm.

After quality phrase mining, a document is transformed from a multiset of words (*bag-of-words*) into a multiset of phrases (*bag-of-phrases*) which will be taken as the input of topic modeling. In this paper, we propose a novel topic model CPhrLDA. Instead of "bag-of-words" assumption which models topic just from the word granularity, CPhrLDA is built on "bag-of-phrases" assumption, and therefore is suitable for phrase-centered topic modeling (discuss in Section 5).

The above stages can mine topical phrases with high frequencies. However, there are some topical phrases that only appear in a certain domain, like `bit vector` appears mostly in the domain of "database". We call such phrases *globally infrequent but locally frequent phrases*. In order to mine such phrases, we use document clustering stage to cluster documents into different domains. And documents within the same domain are used as the new input of the next round to search for domain-specific phrases.

The last three steps of CQMine framework will be iteratively performed until two conditions are satisfied: the difference of cluster assignments between two rounds is less than a certain threshold, or no new phrase could be mined.

We will discuss the three major stages in detail in the following three sections.

4 Quality Phrase Mining

In this section, we firstly describe our phraseness measurement. Then we discuss how to guarantee the completeness requirement. In the third subsection, we discuss our overlapping phrase segmentation method. Finally, we introduce an effective data structure *PhraseTrie* which is used for accelerating the whole process.

4.1 Phraseness Measurement

In order to meet the phraseness requirement of phrase quality criterion, we adopt purely statistics-based measurement χ^2 test [18], which is formalized as:

$$CL_{<Pr_l,Pr_r>} = \sum_{t\in\mathbb{T}} \frac{(O_t - E_t)^2}{E_t} \tag{2}$$

where $CL_{<Pr_l,Pr_r>}$ is the test statistic which measures the statistical possibility of subphrases Pr_l and Pr_r constituting a co-occurring longer phrase $Pr_l \oplus Pr_r$. O_t and E_t denote the observed frequency and expected frequency under condition t, respectively. The distribution of $(Ot - Et)^2$ will follow the χ^2 distribution is based on the assumption that the random variable $(O_t - E_t)$ is independent normally distributed, which is valid due to the central limit theorem.

Let $\neg Pr$ be any phrase other than Pr, there are four conditions in $\mathbb{T}$: $t_1 = Pr_l \oplus Pr_r$, $t_2 = Pr_l \oplus \neg Pr_r$, $t_3 = \neg Pr_l \oplus Pr_r$, and $t_4 = \neg Pr_l \oplus \neg Pr_r$. For example, t_1 means Pr_l and Pr_r co-occurred. The computation of observed frequencies and expected frequencies (i.e., Contigency table) are shown in Table 3.

Given a significant difference (also known as p-value, the probability for a given statistical model that, when the null hypothesis is true, the statistical summary would be the same as or of greater magnitude than the actual observed results) α ($0 < \alpha < 1$ and commonly set as 0.05 [26]), we can define a boolean function to decide whether Pr_l and Pr_r meet the phraseness requirement:

$$\upsilon(Pr_l, Pr_r) = \begin{cases} 1, & if\ CL_{<Pr_l,Pr_r>} \geq \chi^2_{1-\alpha}(1) \\ 0, & otherwise \end{cases} \tag{3}$$

Particularly, we wish to point out that other statistical metrics such as t-test [9] and point-wise mutual information (PMI) can be readily plugged into our framework. We will discuss the performance of our framework under different phraseness measurements in Section 7.

TABLE 3
Contigency Table of Observed Frequencies and Expected Frequencies of Phrase Pr_l and Phrase Pr_r

	Pr_r	$\neg\, Pr_r$
Pr_l	$O_{t_1}: f(Pr)$ $E_{t_1}: f(Pr_l) * f(Pr_r)/N$	$O_{t_2}: f(Pr_l) - f(Pr)$ $E_{t_2}: f(Pr_l) * (N - f(Pr_r))/N$
$\neg\, Pr_l$	$O_{t_3}: f(Pr_r) - f(Pr)$ $E_{t_3}: (N - f(Pr_l)) * f(Pr_r)/N$	$O_{t_4}: N - f(Pr_l) - f(Pr_r) + f(Pr)$ $E_{t_4}: (N - f(Pr_l)) * (N - f(Pr_r))/N$

4.2 Complete Phrase Mining based on q-Chunk

The completeness of extracted phrases highly depends on the merge order. In order to obtain the complete phrases, we need to enumerate every possible merge order. Obviously, a straight-forward algorithm of finding the complete phrases in document d is: enumerating all the subsequences of this document first, then verify whether each one is a complete phrase. The straight-forward algorithm has $O(|d|^4)$ time complexity, since enumerating will generate $|d|^2$ subsequences, and the verification of each subsequence needs $O(|d|^2)$ cost in the worst case.

4.2.1 Dynamic Programming Based Approach

Considering the high time complexity of the above straight-forward algorithm, it is time consuming and cumbersome to perform it especially on a large corpus. According to Definition 4, a phrase is generated by merging two adjacent phrases with high statistical significance. Thus, if $d[i,j]$ is a phrase, then there must be some k ($k \in [i+1, j]$), such that $d[i,k-1]$ and $d[k,j]$ are also phrases. Thus, the problem can be decomposed into a collection of sub-problems. On top of this, we propose a dynamic programming based approach (DPBA) to improve efficiency. To be specific, we set up a matrix G to store the solution of a document, in which a cell $G(i,j)$ stores a boolean value to denote whether $d[i,j]$ is a phrase. To compute each cell, we utilize the phraseness measurement introduced in Section 4.1 to decide if two adjacent sequences $d[i,k-1]$ and $d[k,j]$ could be merged as a longer phrase, note that, $d[i,k-1]$ and $d[k,j]$ must also be phrases first.The recursion formula of DPBA is given in Eq. (4).

$$G(i,j) = \begin{cases} true, & if\ i = j \\ \bigvee_{k=i+1}^{k=j} \bigwedge \begin{pmatrix} G(i,k-1) \\ G(k,j) \\ \upsilon(d[i,k-1], d[k,j]) \\ f(d[i,j]) >= ft \end{pmatrix}, & otherwise \end{cases} \tag{4}$$

In Eq. (4), $\bigvee$ means disjunction and $\bigwedge$ means conjunction.

Lemma 1. *The time complexity of DPBA is $O(n^3)$.*

Proof. In the worst case, the whole upper triangular matrix needs to be computed and kept for back-tracing to find the optimal phrases, and compute each cell needs a cost of $O(n)$, so the time complexity is $O(n^3)$. □

4.2.2 q-Chunk Based Approach

The DPBA algorithm adopts a top-down search strategy, that is, assume the whole sequence is a phrase, then check whether this assumption is valid via computing sub-problems. The advantage of this strategy is that the algorithm can directly return if a primary assumption is satisfied, thus avoid stepping into secondary sub-problems. However, it may cause unnecessary computation when the sequence is long but the phrase is short (since it always check whether the longer sequences are phrases first).

On the contrary, bottom-up search strategy works in a way where small phrases are merged into a longer phrase. This strategy is more suitable for finding small phrases, but as phrases become longer, it needs to repeatedly try to merge the small sequences until it confirms there are no longer phrases, therefore it needs more time to decide whether merging should be terminated.

A fact of phrases is that, commonly, the number of words in a phrase (phrase length) follows a long-tailed distribution, indicating that most of phrases have relatively fixed and short lengths. Based on the fact, we integrate the above two strategies, and propose a more efficient q-chunk based approach (QBA).

The basic idea of QBA is that, firstly, it utilizes top-down search to generate local solutions within a certain length range, then merges these local solutions to generate final results in a bottom-up manner. The main processing steps of QBA are as follows: (1) partitioning the sequence into a series of q-length chunks; (2) performing top-down search on each chunk to get local solutions using Eq. (4); (3) checking whether two adjacent chunks need to be merged. If they do not need to be merged, it means no phrase could cross the boundary between the two chunks. Otherwise the two chunks are merged into a new chunk and QBA will find new solutions on the new chunks. Let $\xi(q)$ be the merge checking cost in step (3). Here we propose two (optional) merge conditions defined on the boundary between two chunks: *exact condition* and *simple condition*. Either of them can be used to check whether two chunks need to be merged.

Exact condition: Given a document d and a boundary position s, for $I = \{i \mid d[i, s-1]$ is a phrase$\}$, $J = \{j \mid d[s, j]$ is a phrase$\}$, if $\{\bigvee_{i\in I}\bigvee_{j\in J} \upsilon(d[i, s-1], d[s, j])\}$ is true, then the two chunks need to be merged.

Lemma 2. *The exact condition is correct.*

Proof. According to Definition 4, any longer phrase is formed by hierarchically merging two sub-phrases. Thus, any l-length phrase $Pr_i = d[i, i + l - 1]$ is formed by $l - 1$ merge operations, and there is one and only one merge operation occurred on each position s within Pr_i ($s \in [i, i + l - 2]$). Therefore, given s is the boundary position of two adjacent chunks, the two chunks need to be merged (i.e., there are possible phrases that could across s) if and only if $\{\bigvee_{i\in I}\bigvee_{j\in J} \upsilon(d[i, s-1], d[s, j])\}$, where $I = \{i \mid d[i, s-1]$ is a phrase$\}$, $J = \{j \mid d[s, j]$ is a phrase$\}$. Thus lemma 2 holds. □

Albeit exact condition could generate fewest chunk merge operations, in worst case, the merge checking cost $\xi(q)$ of a single check incurs $O(q^2)$, since we need to check every possible range between two q-length chunks. Therefore, here we propose an non-exact but more efficient condition to allow a little more candidates, which we call *simple condition*.

Simple condition: Given a document d, and the boundary position s, if $\upsilon(d[s-2], d[s-1]) \vee \upsilon(d[s-1], d[s]) \vee \upsilon(d[s], d[s+1])$ is true, these two chunks need to be merged.

The simple condition is non-exact and therefore it may introduce error. But from empirical analysis, such error is acceptable (less than 1%). The merge checking cost $\xi(q)$ of using simple condition is a constant.

Algorithm 1: QBA

```
   Input: A cluster of documents C, chunk length q
   Output: bag-of-phrases of each document
1  foreach document d_i ∈ C do
2      Initialize matrix G ← φ;
3      Decompose d_i into ⌈|d_i|/q⌉ chunks, and let its
         boundary positions set be S = {s_1, s_2, ..., s_⌈|d_i|/q − 1⌉};
4      DPBA(0, s_1);
5      foreach s_i ∈ S do
6          DPBA(s_i, s_{i+1});
7      DPBA(s_{|S|}, |d|);
8      cur_left ← 0;
9      foreach s_i ∈ S do
10         if s_i satisfies the condition then
11             DPBA(cur_left, s_{i+1});
12             while cur_left ≠ 0 ∧ cur_left satisfies the
                 condition do
13                 cur_left ← cur_left − q;
14                 DPBA(cur_left, s_{i+1});
15         else
16             cur_left ← s_i;
17     PR ← back-tracking on G;
18     Replace original words by complete phrases in PR;
19 return document's bag-of-phrases form
```

The algorithm QBA firstly generates $\lceil \frac{|d_i|}{q} - 1 \rceil$ boundaries S. It then computes the local solution of each chunk using DPBA (line 4-5). We set a variable cur_left (line 8) to denote the left boundary of current chunk. For each boundary s_i, algorithm QBA checks whether s_i satisfies merge condition (line 8). Note that both the above two conditions (i.e., exact and simple condition) can be used here, users can choose any one of them to implement Algorithm 1 based on their intention. If s_i satisfies merge condition, it means that s_i maybe right on a phrase, so Algorithm 1 will conduct a backward search on a new chunk which starts with cur_left and ends with s_{i+1} to include the left part of the phrase in. The backward search ends when cur_left does not need to be merged or reaches the beginning of d, which means the current chunk has included all the left part of the phrase (line 9-11). Otherwise it assigns s_i to cur_left, which means the computation on previous chunk have done and now a new chunk starts with s_i (line 12-13). Finally, a back-tracking process (line 14) is needed to find complete phrases and to replace the original words in d with the newly found phrases (line 15).

Theorem 1. *Given a document d, phrase ratio r (the number of phrases divided by document length), and the chunk length q. Assume the average length of each phrase is l, and the merge checking cost is $\xi(q)$. The expected time cost of Algorithm 1 is $O(n_s q^3 + n_s \xi(q) + n_s p_c \{[(1 + \lceil \frac{l}{q} \rceil)q]^2 + \lceil \frac{l}{q} \rceil \xi(q)\})$, where $n_s = \lceil \frac{|d|}{q} - 1 \rceil$, and $p_c = r \cdot l$.*

Proof. Please refer to Appendix A.2. □

Success in time reduction of QBA hinges to a great extent on the selection of a "good" argument q. Theorem 1 offers us a quantitative way to estimate a theoretically optimal argument q ahead of time given the average phrase length l, the document length $|d|$, and phrase ratio r. Empirically, the values of l and r are relatively fixed, and can be easily estimated

by empirical statistics. Parameter $|d|$ is a known parameter, thus the parameter q can be theoretically estimated as $\arg\min_q f(q) = n_s q^3 + n_s \xi(q) + n_s p_c \{[(1+\lceil \frac{l}{q} \rceil)q]^2 + \lceil \frac{l}{q} \rceil \xi(q)\}$ when simple condition is used.

4.3 Overlapping Phrases Segmentation

In certain documents, complete phrases may overlap with each other. Example 1 shows two paper titles with overlapping phrases.

Example 1. *1) A Non-asymptotic Penalized Criterion for* [*Gaussian Mixture* [***Model***] *Selection*].
2) The Advantages of Abstract [*Control* [***Knowledge***] *Expert* [***System***] *Design*].

In the first title of Example 1, `Gaussian Mixture Model` and `Model Selection` are both complete phrases. But in common sense, word `Model` can only belong to one phrase, i.e., `Gaussian Mixture Model | Selection` or `Gaussian Mixture | Model Selection`. The second title shows a slightly complex case, three complete phrases `Control Knowledge`, `Knowledge Expert System` and `System Design` overlap with each other. This kind of problem is known as segmentation or punctuation problem in grammar. In this paper, we view this problem as a sequence partition problem where each partition denotes a non-overlapping phrase. To guarantee appropriateness, we segment the overlap part of contiguous overlapping phrases into several non-overlapping phrases by probabilistically reasoning about the "optimal" segmentations. Here, we consider the "optimal" should satisfy two requirements: the intra-cooccurrence of phrases and isolation of partition position. For the sake of efficiency and simplicity, we take all contiguous overlapping phrases (like the second title in Example 1) as a whole input instance.

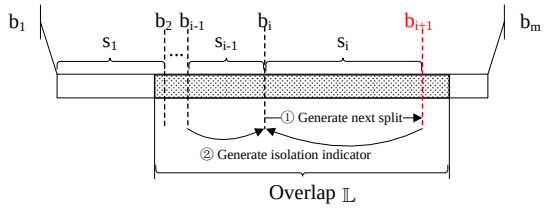


Fig. 2. The two-step generative model of overlapping phrases segmentation in Eq. (1)

Our overlapping phrases segmentation (OPS) problem is formalized in Definition 6. In Definition 6, Eq. (1) is derived from the following two-step generative model (shown in Fig. 2): At the first step, given the previous split position b_i, the model generates split position b_{i+1} in a probability $p(L = b_{i+1} - b_i)$. According to empirical statistic, the random variable L follows a Poisson distribution.

$$p(L) \sim \frac{\lambda^L e^{-\lambda}}{L!}. \tag{5}$$

After generating the sequence length, next, we generate the word sequence itself according to a multinomial distribution over L such that $L = l(s_i)$. Suppose for each subsequence s_j in corpus, we have its frequency $f(s_j)$, the probability can be estimated by the following equation:

$$p(s_i|l(s_i)) = \frac{f(s_i)}{\sum_{\forall l(s_i)=l(s_j)} f(s_j)} \tag{6}$$

Currently, $p(s_i|l(s_i))$ is an unknown parameter, since we have not got the final segmentation result to calculate the exact frequency $f(s_j)$. For learning unknown parameters, existing approaches often employ the Expectation-Maximization (EM) strategy [11], which works in an iterative way, and it would lead to an extremely heavy learning cost. In our method, we utilize the result of complete phrase mining (discussed in previous subsection), the parameter $p(s_i|l(s_i))$ can be estimated without losing too much accuracy (as shown in Theorem 2). Thus, the high parameter learning cost could be avoided to greatly reduce the time cost. Let the actual value of $p(s_i|l(s_i))$ be ϑ, and our estimated value be ϑ', Theorem 2 shows the relative error bound of our estimation.

Theorem 2. *The relative error bound ε of our parameter estimation is $\varepsilon(\rho,\epsilon) \le \max\{\epsilon, \frac{\rho-\epsilon}{1-\rho}\}$, where $\epsilon = \frac{\tau}{\sum_{\forall l(s_j)=l(s_i)} f(s_j)}$, and $\rho = \frac{\tau}{f(s_i)}$, in which τ denotes the total number of overlapping phrases.*

Proof. Please refer to Appendix A.3 □

The first step of the generative model is in accord with $p(s_i, b_{i+1}|b_i)$ in Eq. (1), which can be probabilistically factorized as follows:

$$\begin{aligned} p(s_i, b_{i+1}|b_i) =& p(b_{i+1}|b_i) \cdot p(s_i|b_{i+1}, b_i) \\ =& p(l(s_i)) \cdot p(s_i|l(s_i)) \end{aligned}$$

At the second step, since b_{i+1} has already been generated, and s_i and s_{i-1} have been given as a presupposition, we generate an isolation indicator $p(b_i^*|s_{i-1}, s_i)$ on b_i to check to what extent could b_i segment sequence $S(b_{i-1}, b_{i+1}-1)$ into two semantically independent phrases. Isolation indicator is a probability indicating the extent to placing a "comma" (i.e., split position b_i) here to split the two adjacent sequences s_i and s_{i-1} into two parts. For example, in the first title of Example 1, we prefer the segmentation `Gaussian Mixture Model | Selection`. The isolation indicator can be estimated as:

$$p(b_i^*|s_{i-1}, s_i) = \begin{cases} 1, & i = 0 \ or \ i = |S| \\ \frac{H(s_{i-1})+H(s_i)}{2 \times I(s_{i-1};s_i)}, & otherwise \end{cases} \tag{7}$$

where $H(s) = p(s)\log p(s)$,

$$I(s_{i-1}; s_i) = p(s_{i-1} \oplus s_i) \log \frac{p(s_{i-1} \oplus s_i)}{p(s_i)p(s_{i-1})},$$

where $s_{i-1} \oplus s_i$ represents a merged phrase of consecutive phrases s_{i-1} and s_i. $p(s)$ is defined as

$$p(s) = \frac{f(s)}{\sum_{s' \in U} f(s')},$$

where U is the collection of all phrases.

Different with the existing model [11] which only considers intra-cooccurrence of phrases and regards the generation of segmentations as an independent process. Our methods comprehensively consider both the intra-cooccurrence of phrases and the isolation of partition position. From a technical perspective, the isolation of "current" split position depends on the "future" generated split position. Thus, we need to check every possible new split positions to determine the isolation of current split position, which makes the computation of optimal segmentations very time-consuming.

To address this issue, we adopt a dynamic programming strategy, which is based on an observation that if b_{i+1} and the previous partition position b_i are the optimal position of $S(1, b_{i+1}-1)$, then b_i and b_{i-1} must be the optimal position of $S(1, b_i - 1)$. Thus we set up a two-dimensional matrix MxP, each cell $MxP[b_i][b_{i-1}]$ stores the optimal probability fetched by positions b_i and b_{i-1} on sequence $S(1, b_i - 1)$.

Algorithm 2 shows our *Overlapping Phrases Segmentation* algorithm. Firstly, it initializes the result list N (line 1) and overlapping range (line 2). Then OPS sets the end position as b_{i+1}, and searches the previous position b_i (line 4-10). If

Algorithm 2: OVERLAPPING PHRASES SEGMENTATION(OPS)

Input: overlapping phrases O and matrix MxP
Output: Non-overlapping phrases list N

1 $N \leftarrow \phi$;
2 $\mathbb{L} \leftarrow$ overlapping range in O;
3 $b_{i+1} \leftarrow |O|$;
4 **foreach** $b_i \in \{0, \mathbb{L}\}$ **do**
5 **if** $MxP[b_i]$ *has not been computed yet* **then**
6 OPS(O[0,b_i],MxP);
7 **foreach** $b_{i-1} \in \{0, [\mathbb{L}.start, b_i - 1]\}$ **do**
8 $p \leftarrow MxP[b_i][b_{i-1}] * p(s_i, b_{i+1}|b_i) * p(b_i^*|s_{i-1}, s_i)$;
9 **if** $p > MxP[b_{i+1}][b_i]$ **then**
10 $MxP[b_{i+1}][b_i] \leftarrow p$;
11 $N \leftarrow$ back-tracking on MxP;
12 **return** N

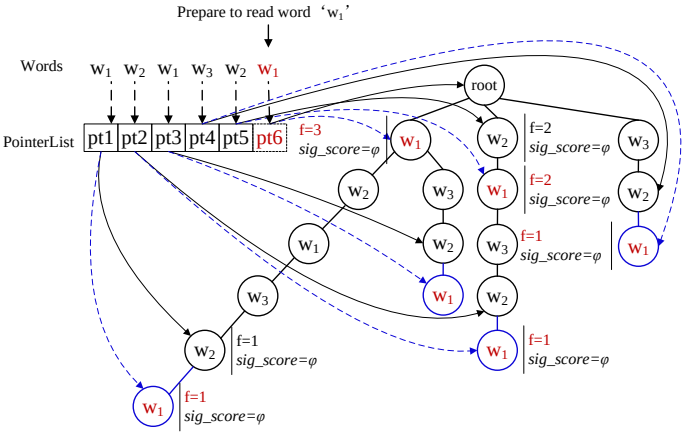


Fig. 3. A running example of inserting a sequence "$w_1w_2w_1w_3w_2$" into a PhraseTrie.

current $MxP[b_i]$ has not been computed, it needs to compute each cell in $MxP[b_i]$ which denotes the optimal segmentation on sequence $S(1, b_i - 1)$ (line5-6). Otherwise, for every possible previous position b_{i-1}, it computes the joint probability p (line 8). If current p is better, it will replaces the original value in $MxP[b_{i+1}][b_i]$(line 9-10). Finally, we use back-tracking(line 11) to find the optimal segmentation.

Theorem 3. *Algorithm 2 is correct.*

Proof. Please refer to Appendix A.4. □

Lemma 3. *The time complexity of Algorithm 2 is* $O(n^3)$.

Proof. Please refer to Appendix A.5. □

Albeit the complexity $O(n^3)$ seems expensive, considering that overlapping phrases only take a small portion of all the phrases (around $1\% \sim 2\%$ in benchmark data) the time cost of the OPS algorithm is acceptable.

4.4 PhraseTrie Building

During the whole process of phrase mining, some basic operations such as phrase frequency counting and frequency retrieval are repeatedly performed. In order to accelerate both frequency counting and retrieval, we design a novel data structure *PhraseTrie* (the right part tree-like structure in Fig. 3). *PhraseTrie* is similar to *trie*, therefore it could store common prefixes to reduce memory usage, also, such a structure could reduce the search space to lower the time cost. Our *PhraseTrie* structure has the same time complexity ($O(n^2)$) with *Hash Counter* in its building. An insert or search operation on a *PhraseTrie* has a fixed time cost $O(|Pr|)$ ($|Pr|$ is the number of words in phrase Pr), while a hash structure cost $O(N)$ time in the worst case (N is total number of words in the corpus and $N \gg |Pr|$), albeit $O(1)$ time in the best case. Moreover, besides counting the frequency of candidate phrases (possible phrases), the *PhraseTrie* structure could also be used for storing the phraseness checking results to avoid repeated computation, and thus reduce the total time cost.

To be specific, every node in *PhraseTrie* stores a word (label), denoting a candidate phrase derived from the root word to the current word, the node also stores the candidate phrase's frequency and its phraseness information (i.e., phraseness state and its significance score). In its building stage, *PhraseTrie* will dynamically set a pointer list, each related with a word in input sequence. Initially, all pointers in the list is pointed to the root node. When inserting a new word, all pointers will be updated by moving to the its child node labeled with the input word. We show a running example in Fig. 3. Assume sequence "$w_1w_2w_1w_3w_2$" has been inserted into the *PhraseTrie*, the black lines shows the states of pointers. Next, word "w_1" needs to be inserted. Each pointer in the list should be updated in the following way: firstly, search whether there is a child node labeled with word 'w_1'. If so, increase the child node's frequency by 1, otherwise, create a new child node with label 'w_1' and set its frequency to 1. Secondly, point it to that child node. The pointers' states after inserting "w_1" is shown in blue lines.

Time complexity analysis of phrase mining stage. Table 4 summarizes the time complexities of phrase mining stage among CQMine and those state-of-the-art methods, where n_d is the number of documents in the corpus, $|d|$ is the average length of a document d.

TABLE 4
Time Complexity Comparison of CQMine with state-of-the-art Methods

CQMine	ToPMine	SegPhrases+	KEA
$O(n_d \cdot \|d\|)$	$O(n_d \cdot \|d\| \cdot log\|d\|)$	$O(iter \cdot n_d \cdot \|d\|^2)$	$O(n_d \cdot \|d\|^2)$

We can see that CQMine has the lowest time complexity compared to state-of-the-art methods.

5 TOPIC MODELING

Traditional topic models either take phrases as unigrams or enforce the words within a phrase to take the same topic assignment. This is usually problematic since a phrase may contain the lexical meaning indicating a different topic from their individual words', especially for idiomatic phrases or terminologies such as "white house" or "max pooling." Therefore, we propose a novel topic model CPhrLDA, which provides a more flexible way that the topic assignments to words of a phrase are not required to be identical.

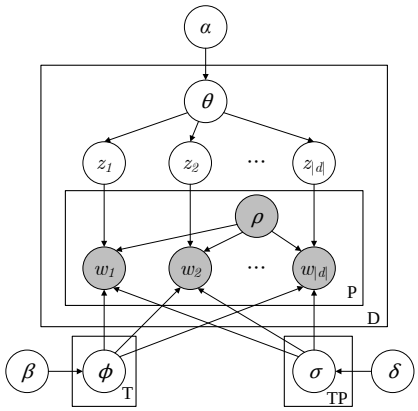


Fig. 4. Bayesian network for CPhrLDA model.

The plate notation of CPhrLDA model is shown in Fig. 4. Let document d contain a set of words $\{w_1, \ldots, w_{|d|}\}$, each

z_i is the topic assignment of w_i, $z_i = z(w_i)$. Variables α, β, and δ are hyper parameters of Dirichlet distributions. θ, ϕ, and σ are three latent variables functioned as parameters of multinomial distributions, which are drawn from their corresponding Dirichlet distributions.

TABLE 5
Notations used in CPhrLDA

Symbol	Description
nt	number of topics
np	number of phrases
nd	number of documents
$\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\delta}$	hyper-parameters of Dirichlet distribution
$\boldsymbol{\theta}^{(d)}$	the topic distribution of document d
ϕ_z	the word distribution of topic z
$\sigma_{z\rho}$	the word distribution of phrase ρ and topic z
$z_i^{(d)}$	the topic of i-th token in the document d
$w_i^{(d)}$	the i-th token in the document d
$\rho(w_i)$	indicate if w_i is a component of a certain phrase
t_j	the j-th topic

Different from the traditional LDA model, we introduce variable $\rho \in \{0, 1\}$ in our CPhrLDA model to determine whether w_i is a component of a phrase. In the stage of quality phrase mining, a phrase has been extracted (see Fig. 1). So, ρ can be observed in the stage of topic modeling. By introducing variable ρ, CPhrLDA model is able to decide whether a word is generated from a topic uni-gram space or from a topic phrase space, which accounts for its better performance on the cases like `max pooling`. The notations used in our model is listed in Table 5.

The generative process of the CPhrLDA model is as follows:

1. Draw $\phi_{t_i} \sim Dir(\beta)$, for each topic $t_i \in \{t_1, \ldots, t_{nt}\}$.
2. Draw $\sigma_{t_i p_j} \sim Dir(\delta)$, for each topic $t_i \in \{t_1, \ldots, t_{nt}\}$ and each phrase $p_j \in \{p_1, \ldots, p_{np}\}$.
3. For each document $d \in \{d_1, \ldots, d_{nd}\}$.
   (a) Draw $\boldsymbol{\theta}^{(d)} \sim Dir(\alpha)$.
   (b) For the k-th word w_x in document d,
      (i) Draw $z_x^{(d)} \sim Multi(\boldsymbol{\theta}^{(d)})$.
      (ii) Draw $w_x^{(d)} \sim Multi(\sigma_{t_i p_j}^{(d)})$, if $\rho(w_x) = 1$;
      draw $w_x^{(d)} \sim Multi(\phi_{t_i}^{(d)})$, otherwise.

We use a posterior distribution $P(\mathbf{w}, \mathbf{z}|\boldsymbol{\rho}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\delta})$ to determine the assignment $\mathbf{z}$ of words $\mathbf{w}$ in a document d. To compute it, we utilize a collapsed Gibbs sampling method [19, 20] to conduct efficient approximate inference. According to the property of Gibbs sampling [19], θ, ϕ, σ can be integrated out, and only the latent variables $\mathbf{z} = z_1, \ldots, z_{|d|}$ are sampled. $P(\mathbf{w}, \mathbf{z}|\boldsymbol{\rho}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\delta})$ can be iteratively calculated according to the following sampling formula. We omit d when d is clear in the content. The details of the Gibbs sampling derivation for CPhrLDA can be found in Appendix A.6.

$$P(t_j|\mathbf{w}, \mathbf{z}(\neg w_i), \boldsymbol{\rho}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\delta}) \propto (\alpha_{t_j} + n_{t_j} - 1) \cdot \begin{cases} \frac{\beta_{w_i} + n_{w_i}^{t_j} - 1}{\sum_{x=1}^{|d|} (\beta_{w_x} + n_{w_x}^{t_j}) - 1}, & if\ \rho(w_i) = 0, \\ \frac{\delta_{w_i} + n_{w_i}^{t_j \rho(w_i)} - 1}{\sum_{x=1}^{|d|} (\delta_{w_x} + n_{w_x}^{t_j \rho(w_i)}) - 1}, & otherwise, \end{cases} \tag{8}$$

where n_{t_j} denotes the number of the event occurrences that any word is assigned to a topic t_j in the document d, i.e. $z(w_i) = t_j$. For any word w_i in d, if it is a unigram, we use $n_{w_i}^{t_j}$ to denote the times that w_i is assigned to the topic t_j; if w_i is a component of any phrase, we use $n_{w_i}^{t_j \rho(w_i)}$ to denote the times that w_i is assigned to t_j.

Time complexity analysis of topic modeling stage. Our CPhrLDA has the same time complexity with the state-of-the-art method ToPMine, both are $O(iter \cdot T \cdot |C|)$, where $|C|$ is the corpus size, T is the number of topics, and $iter$ is the number of iterations.

6 Document Clustering

As we discussed in the CQMine framework, we would like to cluster documents into different domains so that we can find those globally infrequent but locally frequent phrases.

In order to better preserve document's thematic structure, instead of clustering documents directly, we cluster topic distributions θ of documents, which can be generated by CPhrLDA model. Here, $\boldsymbol{\theta} = \{\boldsymbol{\theta}^{(d_1)}, \boldsymbol{\theta}^{(d_2)}, ..., \boldsymbol{\theta}^{(d_{nd})}\}$, where $\boldsymbol{\theta}^{(d_i)} \in \mathbb{R}^{nt \times 1}$ is the topic distribution of document d_i, and it can be computed using posterior estimating: $\boldsymbol{\theta}^{(d_i)} = [\theta_{t_1}^{(d_i)}\ \theta_{t_2}^{(d_i)}\ \ldots\ \theta_{t_{nt}}^{(d_i)}]$, where each element $\theta_{t_j}^{(d_i)} = \frac{n_{t_i} + \alpha_{t_j}}{\sum_{x=1}^{nt} n_{t_x} + nt \cdot \alpha_{t_x}}$.

We adopt an improved k-means clustering method, density peak based k-means (DPBK) [21] to cluster $\boldsymbol{\theta}$ into k ($k < n$) clusters $C = \{C_1, C_2, ..., C_k\}$ by minimizing the following total divergence:

$$TotalDivergence = \sum_{i=1}^{k} \sum_{\boldsymbol{\theta}^{(d_j)} \in C_i} JSD(\boldsymbol{\theta}^{(d_j)}, c_i), \tag{9}$$

where c_i is the arithemetic average centroid of C_i, and function JSD is the Jensen-Shannon divergence:

$$JSD(\boldsymbol{\theta}^{(d_i)}, \boldsymbol{\theta}^{(d_j)}) = \frac{1}{2} KL(\boldsymbol{\theta}^{(d_i)}, \boldsymbol{\theta}^{(d_j)}) + \frac{1}{2} KL(\boldsymbol{\theta}^{(d_j)}, \boldsymbol{\theta}^{(d_i)}), \tag{10}$$

where $KL(\boldsymbol{\theta}^{(d_i)}, \boldsymbol{\theta}^{(d_j)})$ is Kullback-Leibler Divergence [22] between $\boldsymbol{\theta}^{(d_i)}$ and $\boldsymbol{\theta}^{(d_j)}$.

DBPK selects initial centroid based on two intuitions: (1) cluster centroids are characterized by higher densities (higher score of Eq.(11)) than their neighbors; (2) centroids are expected to have a relatively large distance from points with higher densities [23] (higher score of Eq. (12)). The $\boldsymbol{\theta}^{(d_i)}$'s local density ψ_{d_i} and relative distance ζ_{d_i} by equations Eq. (11) and Eq. (12), respectively.

$$\psi_{d_i} = \sum_{j=1}^{nd} \varphi(JSD(\boldsymbol{\theta}^{(d_i)}, \boldsymbol{\theta}^{(d_j)}) - r), \tag{11}$$

where $\varphi(x) = 1$ if $x < 0$ and $\varphi(x) = 0$ otherwise, and r is a cut-off distance which is defined as:

$$r = \frac{1}{\mu^{|\max_{i \in [1,nt]}(\alpha_{t_i}) - 1|}},$$

where $\mu > 1$ is a user specified constant coefficient.

The relative distance of $\boldsymbol{\theta}^{(d_i)}$ is measured by the minimum JSD distance between $\boldsymbol{\theta}^{(d_i)}$ and any other $\boldsymbol{\theta}$ with higher density than $\boldsymbol{\theta}^{(d_i)}$:

$$\zeta_{d_i} = \min_{j: \psi_{d_j} > \psi_{d_i}} JSD(\boldsymbol{\theta}^{(d_i)}, \boldsymbol{\theta}^{(d_j)}), \tag{12}$$

Incorporating these two factors, the final score of each $\boldsymbol{\theta}^{(d_i)}$ is defined as Eq. (13):

$$Score(\boldsymbol{\theta}^{(d_i)}) = \zeta_{d_i} \times \psi_{d_i}. \tag{13}$$

We rank $\boldsymbol{\theta}$ according to their scores computed by Eq. (13), and take the top-k points as the initial centroids. Then, we utilize k-means clustering to clustering $\boldsymbol{\theta}$ until all $\boldsymbol{\theta}$ have been assigned to their nearest centroids.

7 Experimental Evaluation

In this section, we evaluated the effectiveness of our CQMine framework by several interpretability tasks, what's more, we conducted efficiency evaluations to demonstrate both high effectiveness and high efficiency of our method over state-of-the-art models.

7.1 Datasets

We chose four real-world datasets, the detailed statistics are summarized in Table 6 (1) **5Conf**[3] is a collection of paper titles of five fields: AI, DB, DM, IR, and ML; (2) **APNews**[4] contains 106K TREC AP news articles (1989); (3) **Titles**[5] is the entire paper titles extracted from DBLP dataset; and (4) **Abstracts**[6] contains 529K computer science paper abstracts from DBLP dataset.

TABLE 6
Statics of Four Datasets We Used as Corpus

Datasets	5Conf	APNews	Titles	Abstracts
# of Documents	44K	106K	1.6M	529K
Vocabulary Size	5K	170K	96K	135K
# of Characters	2.8M	229.9M	186.6M	478.8M

7.2 Compared Methods

To demonstrate the interpretability and efficiency of our proposed method, we compared our method with the following six state-of-the-art methods.

- **ToPMine**[9] is a scalable method for unsupervised topical phrase mining from big text corpora.
- **CITPM**[21] is an improved version of ToPMine, it adopts a statistics-based method to check phraseness and an iterative framework to mine phrases.
- **SegPhrases+**[11] is a scalable method for mining quality phrases from massive text corpora. It filters overestimated phrases and refines quality phrase with a segmentation-based strategy.
- **C-value**[24] utilizes NLP chunking to obtain candidate phrases and uses C-value to refine quality phrases. We implemented it using Stanford NLP[7] and NLTK language toolkit[8].
- **KEA**[25] is a supervised keyphrases (multi-word units) extraction algorithm from text documents.
- **spaCy**[9] is an industrial-strength NLP system. We used its noun chunks API to extract phrases.

7.3 Experimental Settings

In our experiments, we adopted the widely used Fisher's standard p-value $\alpha = 0.05$ for all datasets, which requires a less than 5% theoretical error between the hypothesis and actual results in statistics [26]. According to Fisher's statement [27], if p-value is between 0.1 and 0.9, there is certainly no reason to suspect the tested hypothesis. If it is below 0.02, it is strongly indicated that the hypothesis fails to account for the facts. Thus, Fisher supported a p-value of 0.05 as an indicating evidence against the null hypothesis.

In cohesion evaluation, we set frequency threshold ft to be 5, 5, 10, 50 for 5Conf, AP news, Titles, and Abstracts dataset, respectively. In phrase quality evaluation, we tuned a wide rang of ft values to comprehensively evaluate the effectiveness since ft is one of the most important parameters of our phrase mining algorithms. We will discuss the setting of ft in Section 7.4.3.

In the complete phrase mining stage, we dynamically estimated an optimal q using the extreme points of the cost function $f(q)$ (see Section 4.2.2), namely $\arg\min_q f(q)$. Notice that, this calculation depends on the average phrase length l and the phrase ratio r, which were determined through empirical statistic. For the four datasets, both l and r are kept stable, e.g. phrase lengths l were around $2.09 \sim 2.35$ and phrase ratios r were around $0.16 \sim 0.25$. Therefore, we adopt $l = 2.1$ and $r = 0.2$ in our experiments. For QBA, we used simple condition as our merge checking condition.

In topic modeling stage, the number of topics $nt = 20$ and the Gibbs sampling iterations was set to 100 for all methods.

To determine the number of document clusters k, there have been many approaches developed. The *rule of thumb* [28] states that the number of clusters has a positive-correlation with the number of documents, i.e., $k \approx \sqrt{n/2}$ (n is the number of documents). Other researches [29–31] propose the *number-of-clusters hypothesis*, stating that the number of clusters within a text database should be large if individual documents are dissimilar, and small otherwise. Another factor is the average document length, for datasets with short document length, e.g., Titles, too many clusters may impair the effectiveness of topic model (either LDA or CPhrLDA) to lower down the accuracy of clustering. By comprehensively considering the number of documents, dissimilarity between documents, and average document length, in this paper, we fixed M at: 10, 10, 50, and 100 for 5Conf, APNews, Titles, and Abstracts dataset, respectively. Other parameters required by state-of-the-art methods were set according to the open source tools' default settings or their reports in literature.

The spaCy was implemented using CPython 2.6+/3.3+. Other methods were implemented using Java JDK 8. The execution time experiments were all conducted on an IBM BladeCenter with an Intel Xeon CPU X5680 v6@3.3GHz and 24GB memory, running Ubuntu (Linux) operating system.

7.4 Interpretability Evaluation

In Section 7.4.1, we conducted topical phrase cohesion and quality evaluation to demonstrate the effectiveness of the whole CQMine framework. Moreover, we compared CPhrLDA model with LDA. In Section 7.4.2 we examined the effectiveness of phrase mining stage. Also, we discussed the setting of frequency threshold and the effect of iteration scheme in Section 7.4.3 and Section 7.4.4 respectively. Finally, a case study shows the capability of CQMine in a visible way (please refer to Appendix B.1).

7.4.1 Topical phrase cohesion and quality evaluation

The cohension and quality of topical phrases are evaluated by a domain expert evaluation and a *phrase intruder* [32] test.

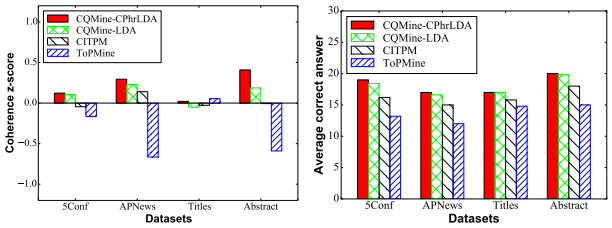
We conducted domain expert evaluation to evaluate both topical coherence and phrases quality. We sorted mined topical phrases by their topical frequencies in descending order, and asked five experts (year-2-above Ph.D. candidates of computer science) to rate the first 30 phrases on a scale of 1 to 10 based on two criteria: (1) whether they are natural, meaningful, complete and (2) whether the topic is consistent with its topic's thematic structure. A phrase is rated 10 if it fully satisfies both criteria, otherwise it gradually decreases to 1. Each expert's rating is transformed to a standard score (z-score), i.e., $zscore = \frac{x-\mu}{\sigma}$, where x is the rating of each expert, μ and σ is the average rating and standard deviation of all experts', respectively. The average score of five experts is regarded as the final score of each method.

Fig. 5(a) shows the coherence value z-score. We can see that CQMine outperforms CITPM and ToPMine for all datasets except Titles dataset. On Titles dataset, there is no significant difference among all methods. The reason is that the document lengths in Titles are very short, both LDA and CPhrLDA would be introduced errors that caused by data sparsity problem. Even though CQMine performs sightly worse than ToPMine, it is hard to say which one is actually better since ToPMine performs only 0.09% better on experts' average rating than CQMine that they are almost on the same level.

In phrase intruder [32] test, each question consists of five phrases, in which four of them were randomly chosen from

3. http://web.engr.illinois.edu/ elkishk2/
4. http://www.ap.org/
5. http://dblp.uni-trier.de/db/
6. http://dblp.uni-trier.de/db/
7. http://nlp.stanford.edu/software/tagger.html
8. http://www.nltk.org/
9. https://spacy.io/

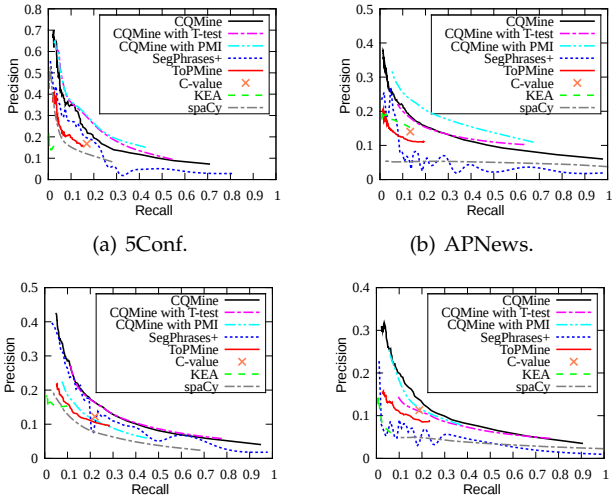
the top-10 phrases (i.e., the phrases with highest frequencies) within one topic, and the remaining one was randomly chosen from the top-10 phrases in a different topic as an *intruder phrase*. We asked five experts to independently select the right *intruder* or to indicate that they are unable to make a choice. The experts could base on their own background knowledge to judge the goodness of phrases or use existing tools (such as search engines, domain knowledge-bases) to facilitate understanding the phrases that they were not familiar with. As seen in Fig. 5(b), CQMine beats ToPMine and CITPM in this task on all datasets.



(a) Domain expert evaluation. (b) Phrase intruder tests.

Fig. 5. Topical phrases cohesion evaluation. Domain expert evaluation shows the standard z-scores rated by five experts' on topical cohesion. Phrase intruder tests shows the average correct answers of 30 phrase intruder questions.

We also compared CPhrLDA with LDA in domain expert evaluation and phrase intruder tests. For both CPhrLDA and LDA, we used the same phrase set mined by our quality phrase mining stage as the input. The result shows in Fig. 5. From Fig. 5, we can find that, CQMine with CPhrLDA outperforms CQMine with LDA in both tasks. To be specific, CPhrLDA achieves a better z-score in domain expert evaluation, and up to 3.1% better average correct answer than LDA in phrase intruder tests.



(a) 5Conf. (b) APNews.

(c) Titles. (d) Abstracts.

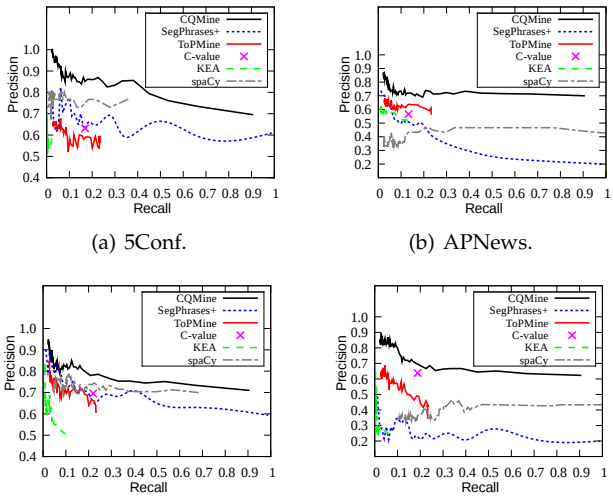
Fig. 6. Precision-recall curves of all methods evaluated by Wiki phrases benchmark on four datasets.

7.4.2 Effectiveness of quality phrase mining stage

We examined the effectiveness of our quality phrase mining stage by measuring the phrase quality in two metrics: (1) *Wiki-phrases* benchmark [11] and (2) Expert Evaluation.

Wiki-Phrases: *Wiki-phrases* is a collection of popular mentions of entities by crawling intra-Wiki citations within Wiki content. *Wiki phrases* benchmark provides a good coverage of commonly used phrases which could avoid the variance caused by different human raters. In this evaluation,we regarded *Wiki phrases* as ground truth phrases, that is to say, there are only two classes: belongs to/not belongs to Wiki phrases. To compute precision, only the *Wiki phrases* are considered to be positive. For recall, we firstly merged all the phrases returned by all methods including ours, then we obtained the intersection between the *Wiki phrases* and the merged phrases as the evaluation set.

In order to test the effectiveness of our framework under different phraseness measurement, besides χ^2-test used in CQMine, we also tested two commonly used metrics: t-test [9] (denotes as CQMine with T-test), and point-wise mutual information (denotes as CQMine with PMI).



(a) 5Conf. (b) APNews.

(c) Titles. (d) Abstracts.

Fig. 7. Precision-recall curves of all methods evaluated by Expert Evaluation on four datasets.

Precision-recall curves (PR-curves) of different methods evaluated by *Wiki-phrases* are shown in Fig. 6. The PR-curves are created by plotting precision against recall at various frequency thresholds ft (from 2 to 150). From Fig. 6, we can see that: firstly, among the three phraseness measurements (i.e., χ^2-test, t-test, and PMI), χ^2-test and t-test achieved similar performance in all four datasets, while PMI achieved a higher F1-score in APNews dataset but lower in Titles dataset. Considering that χ^2-test achieved better precision or recall than the other two metrics, it would be the best choice among the three; Secondly, comparing CQMine (with t-test) with ToPMine which also adopts t-test as its phraseness measurement, CQMine (with t-test) outperforms ToPMine on all four datasets, indicating our quality phrase mining algorithms could overcome order sensitive and inappropriate segmentation problems that brought by traditional approaches; Finally, comparing all three versions of CQMine with other methods, except on Titles dataset where CQMine with PMI performs slightly worse than SegPhrase+, all CQMine methods excels significantly. To be specific, CQMine could achieve a higher recall with the same precision, or a higher precision with the same recall. Another observation is that the curves of CQMine are more stable than SegPhrases+, indicating our method is less sensitive to frequency threshold than SegPhrases+. Comparing with the NLP-based method spaCy, CQMine excels significantly. The main problem of spaCy is that most of the phrases it extracted are less informative (violating phraseness criterion), e.g., "this market", "such a image". This demonstrates that the phrase quality of statistic-based methods could be better than NLP-based methods. In Fig. 6 and Fig. 7, some curves terminate at very low recall level, the reason is these methods extract smaller number of quality phrases than others . One example is KEA, which only extracts few "key phrase".

Expert Evaluation: We also conducted an expert evaluation to examine whether the phrases returned by different

methods are natural, meaningful, complete or not. There are only two classes in expert evaluation: positive/negative phrases.Expert Evaluation provides a good contrast with Wiki-phrases benchmark, which is regarded as "low bias but high variance," while Wiki-phrases Evaluation is regarded as "high bias but low variance." For each dataset, we randomly sampled 500 *Wiki-phrases* uncovered phrases and asked five experts to mark positive and negative phrases. We varied ft from 2 to 150 and the experimental results using Expert Evaluation are shown in Fig. 7.

Comparing with Fig. 6 and Fig. 7, the advantages of CQMine are more significant on the Expert Evaluation. The reason is that, *Wiki-phrases* do not completely cover all phrases especially for some terminologies. This is also an import reason to conduct the Expert Evaluation. From Fig. 7, we can see that CQMine has a significant advantage compared with other methods. To be specific, CQMine can achieve higher precisions and recalls. Also, when CQMine achieved the maximum recall, its precision still maintained at a fairly high level. The evaluation results suggest that our method is not only able to extract commonly used phrases that listed in *Wiki phrases*, but also work properly for long tail terminologies which might occur not so frequently.

We also conducted an empirical test using Wiki-phrases benchmark to validate the coverage of the pair-wised merging property used in DPBA. We defined coverage ratio as the proportion of the number of phrases that meet this property to the total number of phrases. Table 7 shows that most of phrases can be extracted using this property.

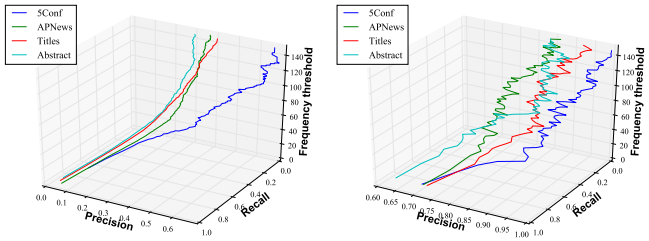
TABLE 7
Coverage ratio of the pair-wised merge property in Definition 4 on four datasets.

	5Conf	APNews	Titles	Abstract
Coverage ratio	$\geq 0.71\%$	$\geq 0.98\%$	$\geq 0.95\%$	$\geq 0.90\%$

7.4.3 Sensitive Analysis of frequency threshold

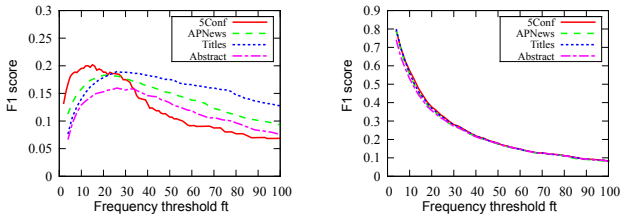
In order to select a proper value for ft, from a statistics perspective, the theoretical minimal sample size requires frequency threshold $ft \gg v + 1$, where v is the number of independent variables. And the conventional rule of thumb [33, 34] for χ^2 test states that the minimal frequency should be greater than 5.

Based on the experimental results shown in Figs. 6 and 7, we use Fig. 8 to show CQMine's precision and recall under different ft values. From Fig. 8 we can see that, in general, high precision requires high ft and high recall requires low ft. The F1-scores of CQMine w.r.t. different ft are shown in Fig. 9. The $F1$-scores increase to some peak values and then drop slowly when using Wiki-phrases benchmark, and the best ft should be $ft \in [10, 30]$. Using expert evaluation, the $F1$-scores form a monotonic curve as increasing ft and the minimal ft value ($ft = 4$) achieves the best $F1$-score. The reason why the two evaluations have different trends is that, in expert evaluation, even at the minimal frequency threshold ($ft = 4$), our method could achieve high enough precision. With the increasing of ft, the number of extracted phrases will be reduced, therefore the $F1$-scores will drop sharply with the reduction of recall. In Wiki-phrases benchmark, since Wiki-phrases is not a complete resource, some terminologies are not covered, which will be regarded as negative phrases and as a consequence leading to the low precision when ft is low. With the increasing of ft, some Wiki uncovered phrases (whose frequency is low) are eliminated and the precision will increase. At the initial period ($ft \in [4, 30]$), the precision increment dominates recall reduction, while after certain ft ($ft > 30$), the recall reduction dominates, therefore, the $F1$-scores will increase to some peak values and then drop slowly when using Wiki-phrases benchmark. In summary, by comprehensively considering the statistical requirement and the empirical evaluation, $ft \in [5, 30]$ would be a good choice for frequency threshold.



(a) Wiki phrases Benchmark. (b) Expert Evaluation.

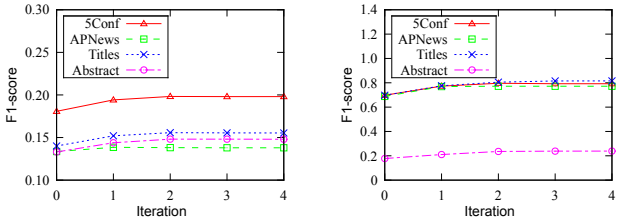
Fig. 8. The curves of precision, recall and frequency threshold ft on Wiki phrases benchmark and expert evaluation.



(a) Wiki phrases Benchmark. (b) Expert Evaluation.

Fig. 9. CQMine's $F1$-scores with various frequency threshold ft on Wiki phrases benchmark and expert evaluation.

In Expert Evaluation, even at the minimal frequency threshold setting ($ft = 4$), our method could achieve a high enough precision (around $0.7 \sim 0.8$, please refer to Fig.7). As the increasing of ft, the number of extracted phrases will be reduced (considering that the phrase frequency follows a long-tail distribution like the shape of Fig. 9 (b)), therefore the $F1$-scores will drop sharply with the reduction of recall. In Wiki-phrases benchmark, since Wiki-phrases is not a complete resource, some terminologies (i.e., the long-tail phrases with low frequencies) are not covered, which will be regarded as negative phrases and as a consequence leading to the low precision when ft is low (please refer to Fig.6). With the increasing of ft, some Wiki uncovered phrases (whose frequency is low) are eliminated and the precision will increase. At the initial period ($ft \in [4, 30]$), the precision increment dominates recall reduction, while after certain ft ($ft > 30$), the recall reduction dominates, therefore, in Fig. 9 (a), the $F1$-scores will increase to some peak values and then drop slowly. We added the discussion into Section 7.4.3 to make it clear.



(a) Wiki-phrases Benchmark. (b) Expert Evaluation.

Fig. 10. CQMine's F1-scores with various number of iterations on Wiki phrases benchmark and expert evaluation.

7.4.4 Effectiveness of iteration scheme

We also verified the effectiveness of our iteration scheme. Fig. 10 shows the results.

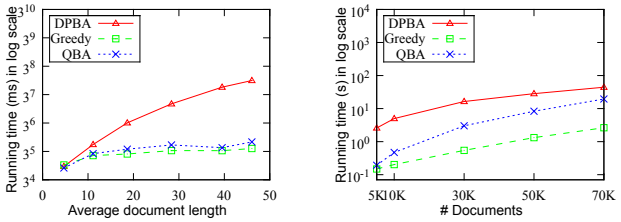
Compared with no iteration, the iteration scheme could improve $F1$-scores on all datasets. Another observation is that, generally, $F1$-scores keep increasing with the growing

number of iterations, while after 2 or 3 iterations, the $F1$-scores almost remain unchanged. Therefore, having multiple iterations is better than having one, and the most proper number of iterations would be 2 or 3.

There are at least four reasons why CQMine outperforms other methods in aspect of interpretability. Firstly, we adopt a statistic-based phraseness measurement method, it could find those phrases which only have statistical significance but with a low frequency. This is more sensible than frequency-based methods like SegPhrases+. On top of that, our complete phrase mining algorithms could strictly guarantee completeness requirement. Thirdly, CPhrLDA incorporates "bag-of-phrases" constraint, and it could address the collocation phrase issue. Finally, CQMine adopts an iterative scheme to mine domain-specific terminologies, which helps both phrase mining and topical inferring.

7.5 Efficiency Evaluation

We firstly tested the efficiency of DPBA, QBA, and a greedy algorithm (Greedy) used in ToPMine by investigating the influence of data size and average document length on running time, as shown in Fig. 11.



(a) Running time varies with average document length. (b) Running time varies with data size.

Fig. 11. Efficiency evaluation for CQMine's complete phrase mining algorithms

To test the influence of average document length, we created a set of synthetic datasets, which have a constant data size ($2.8MB$) but different document lengths (from 4.69 to 46.06). Fig. 11(a) shows that, the running time of DPBA is very sensitive to the growing of average document length, whereas Greedy and QBA are less sensitive to the average document length. QBA could achieve almost the same running time as Greedy since it adopts an efficient partition-based strategy, even though the Greedy has an $O(nlogn)$ computational complexity.

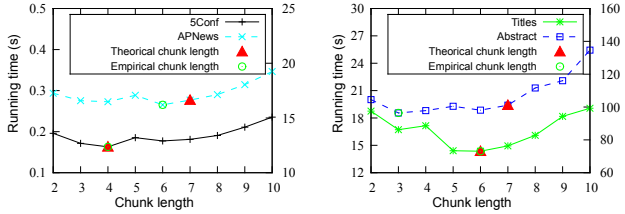


Fig. 12. Running time of CQMine with different chunk length q. The running time of CQMine with theoretically estimated chunk length are also marked.

We also varied data size from 5K to 70K (average document length is 40). Fig. 11(b) shows that all methods take more time in a larger data size. Compared with QBA, the running time of DPBA grows faster. This demonstrates that the partition-based strategy adopted in QBA could greatly reduce time cost and make efficiency competitive even to the greedy algorithm.

Fig. 12 shows the effectiveness of our chunk length estimation method. As discussed in Section 4.2.2, we hope to use a "good" chunk length to save running time. In Fig. 12, the empirically optimal chunk length is the one with the minimal running time (denoted with circle) and the theoretically estimated chunk lengths are denoted with Δ symbol. We can see that the theoretically estimated chunk length was equal to the empirically optimal chunk length on 5Conf and Titles datasets, respectively. For APNews dataset, the estimated chunk length was $q = 7$, and the running time under this chunk length was $16.637s$. The empirically optimal chunk length was $q = 6$ with a running time of $16.202s$. The time difference between estimated optimum and empirical optimum was only $0.435s$. On Abstract dataset, our estimated chunk length was $q = 7$ with a running time of $101.139s$, and the empirical best chunk length was $q = 3$ with a running time $96.414s$. The time difference between them was only $4.725s$. It demonstrated that our estimated chunk lengths were close to the empirical results.

TABLE 8
Running time comparison of CQMine's quality phrase mining stage with state-of-the-art methods

Methods \ Datasets	5Conf	APNews	Titles	Abstracts
CQMine	0.35s	32s	57s	4min01s
ToPMine	5.43s	4min25s	5min6s	9min34s
SegPhrases+	9.02s	19min14s	25min56s	54min4s
KEA	63s	12min9s	12min35s	32min49s
C-value	11.53s	3min50s	13min82s	10min30s
spaCy	7.91s	12min33s	9min13s	24min59s

The practical execution time of CQMine's quality phrase mining stage (with QBA) compared with state-of-the-art methods on different datasets are shown in Table 8. As expected, the utilization of *PhraseTrie* structure and the efficient complete phrase mining methods make CQMine achieve better efficiency. We can find that CQMine have a huge advantage compared with SegPhrases+ ($13.5\times \sim 36\times$ faster) and KEA ($8.2\times \sim 180\times$ faster). Our statistics-based method has a huge advantage compared to NLP-based methods. Even though spaCy claimed it is the fastest NLP-based system, our method still achieved a better efficiency ($6.2\times \sim 23.5\times$ faster). Compared with ToPMine, CQMine also achieved a faster running time ($2.4\times \sim 15.5\times$ faster).

TABLE 9
Running time of different components in CQMine's quality phrase mining stage on four datasets.

Component \ Datasets	5Conf	APNews	Titles	Abstracts
Frequency counting	0.127s	13.204s	33.687s	89.621s
Phrase mining	0.167s	14.744s	15.333s	101.139s
Overlap segmentation	0.056s	4.257s	8.165s	49.923s

Table 9 shows each component's time cost in quality phrase mining stage (with QBA). One observation is that, although our overlapping phrases segmentation algorithm has an $O(n^3)$ time complexity, it did not occur too much overhead since overlapping phrases only take a small portion of all the phrases.

TABLE 10
Running time of different stage in CQMine on four datasets.

Stage \ Datasets	5Conf	APNews	Titles	Abstracts
Preprocessing	3s	92.065s	147s	199.925s
Complete Phrase Mining	0.31s	34s	56s	240s
Topic Modeling	2.523s	196.721s	663.168s	632.519s
Document Clustering	3.65s	57.867s	106.147s	288.799s
Overall	9.483s	380.653s	972.315s	1361.243s

Table 10 shows the total running time of CQMine along with the specific time cost in each stage of a single iteration.

The above two sets of experiments demonstrate CQMine could achieve a higher efficiency in both phrase mining

and topical phrase mining than state-of-the-art methods. Empirically, our method not only is the one with the best phrase quality but is the fastest one as well.

8 Related work

The importance of phrase mining has led to a substantial amount of research over the past few decades.

8.1 Phrase mining

Phrase mining problem originates from the noun phrase chunking or NP-chunking problem in NLP community. Originally, there have been many NP-chunking methods [4, 5], most of which rely on part-of-speech (POS) tagging information, and use predefined chunk grammar rules to create NP-chunker. However, these chunk grammar based methods often produce less informative phrases. Therefore, supervised NP-chunking methods [35, 36] or stochastic-based methods [37–40] have been developed to improve accuracy. Supervised NP-chunking methods use annotated texts to train classifier-based chunkers on top of a variety of additional features. However, supervised methods may suffer from high annotation cost since to obtain hundreds of manually annotated training texts is expensive. Stochastic-based methods use stochastic models, such as HMM model [38] and CRF [39, 40] model to parse noun phrases. However, these methods suffer from low scalability to new languages, new domains or genre. These shortages refrain them from domain-specific, dynamic, grammar-free texts such as twitter, academic paper and query logs.

A recent trend is to leverage distributional features derived from the big corpus to further improve phrase quality. Pitler [15] uses web-scale corpus's distributional features and adopts a statistical metrics PMI to mine n-grams. Parameswaran [8] uses several indicators to extract n-grams. Deane [10] proposes a statistical measure based on Zipfan ranks to measure lexical association in a phrase. The statistic-based methods can achieve higher scalability than aforementioned methods, since they do not rely on language-specific linguistic features. However, these methods suffer from order sensitive problem.

Word sequence segmentation (or phrasal segmentation) is another strategy for phrase mining. Formally, phrasal segmentation aims at partitioning a word sequence into a set of disjoint subsequences, each indicating a phrase. A recent work is SegPhrases+ [11]. It only considers intra-cooccurrence of phrases such as phrase length and words, while ignores the inter-isolation between phrases.

In this paper, we propose a new phrase mining approach. Our approach could eliminate order sensitive and inappropriate segmentation, so that it could achieve a better accuracy than existing methods.

8.2 Topical phrase mining

Significant progresses has been made on the topical phrase mining and they can be broadly classified into three types: (1) joint learning phrases and their topic assignment, (2) mining phrases posterior to topic inferring, and (3) mining phrases prior to topic inferring.

For the first strategy, it performs phrase mining and topic inferring simultaneously by incorporating successive word sequence assumption into the generative model. Wallach [41] proposed a bigram topic model based on a hierarchical Dirichlet allocation model. Bigram model is a probabilistic generative model that conditions on the previous word and topic when drawing the next word. Wang [12] proposed a topical n-gram model that infers n-grams by concatenating successive bigrams. Lindsey [14] proposed a PDLDA model, a hierarchical generative model assuming that the probability of a next Bayesian change-point depends on the current topic and word. In these models, whether two consecutive words can be formed to a bi-gram depends on the occurrences of the front word and its topic assignment, which would make them easy to generate less informative phrases. Another main shortage is that, these methods may suffer from high model complexity, which may generally result in overfitting on training data and demonstrate poor scalability outside small datasets [9].

The second strategy utilizes a post-processing step to generate phrases after inferred by the LDA model. TurboTopics [42] recursively merges consecutive words with the same latent topic by a distribution-free permutation test on arbitrary length back-off model until all significant consecutive words have been merged. KERT [6] performs frequent pattern mining on each topic as a post-processing step to LDA. This strategy may encounter the collocation problem where unigrams in different topic cannot be aggregated to form a phrase especially for idiomatic phrases.

The third strategy is mining phrases prior to topic inferring. It was first proposed by ElKishky [9], It firstly performs frequent contiguous pattern mining to find candidate phrases, then refines candidates by merging adjacent unigrams and then transforms original documents into *bag-of-phrases*, and finally, uses an improved LDA to infer topical phrases.

In this paper, we propose a novel topical phrase mining method CQMine. Our method could achieve a better performance than state-of-the-art methods in terms of phrase quality and topical cohesion.

9 Conclusion

We presented an efficient method for cohesion and quality topical phrase mining. In phrase mining stage, we focus on quality phrase mining problem, and propose two efficient quality phrase mining algorithms. In practice, the time cost of our best exact algorithm is competitive to greedy algorithm. In topic modeling stage, we propose a novel topic model to incorporate the constraint that is induced by phrases, moreover, it can well address the collocation phrase issue. Finally, considering the fact that some phrases are only valid in certain domains, we cluster documents under the condition that they share similar topic distribution and iteratively perform cluster updating and topical inferring to further improve the cohesion of topical phrases. The empirical verification demonstrated our framework has high interpretability and efficiency.

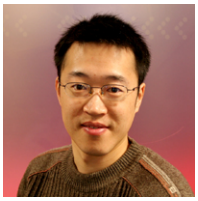
References

[1] Leskovec J, Backstrom L, Kleinberg J., "Meme-tracking and the dynamics of the news cycle," in ACM SIGKDD, 2009, pp.497-506.
[2] Li M, Wang J, Tong W, et al., "EKNOT: Event Knowledge from News and Opinions in Twitter," in AAAI , 2016.
[3] He Z, Chen C, Bu J, et al., "Document Summarization Based on Data Reconstruction," in AAAI , 2012.
[4] S.P. Abney, "Parsing by Chunks," In Principle-based parsing, 1991, pp.257-278.
[5] Clahsen H, Felser C., "Grammatical processing in language learners," Applied Psycholinguistics, 2006, 27(27):3-41.
[6] Danilevsky, M., Wang, C., Desai, N., et al., "Automatic construction and ranking of topical keyphrases on collections of short documents'" In SDM, 2014.
[7] Simitsis A, Baid A, Sismanis Y, et al., "Multidimensional content exploration," in VLDB, 2008, 1(1):660-671.
[8] Parameswaran A, Garcia-Molina H, Rajaraman A., "Towards the web of concepts: Extracting concepts from large datasets," in VLDB, 2010, 3(1-2):566-577.
[9] El-Kishky, A., Song, Y., Wang, C., et al., "Scalable topical phrase mining from text corpora," In VLDB, 2014, 8(3):305-316.
[10] Deane P., "A nonparametric method for extraction of candidate phrasal terms," in ACL, 2005, pp.605-613.
[11] Liu J, Shang J, Wang C, et al. Mining quality phrases from massive text corpora," in ACM SIGMOD, 2015, pp.1729-1744.

[12] Wang, X., McCallum, A., Wei, X., "Topical n-grams: Phrase and topic discovery, with an application to information retrieval," In ICDM, 2007, pp. 697-702.
[13] Wang, C., Danilevsky, M., Desai, N., et al., "A phrase mining framework for recursive construction of a topical hierarchy," In ACM SIGKDD, 2013, pp. 437-445.
[14] Lindsey, R. V., Headden, III W. P., Stipicevic, M. J., "A phrase discovering topic model using hierarchical pitman-yor processes," In EMNLP, 2012, pp. 214-222.
[15] Pitler E, Bergsma S, Lin D, et al. Using web-scale N-grams to improve base NP parsing performance. In ACL ICCL, 2010: 886-894.
[16] Porter, M. F., "Snowball: A language for stemming algorithms," Open Source Initiative Osi, 2001.
[17] Porter, M. F., "An algorithm for suffix stripping," Program, 1980, 14(3):130-137.
[18] Pearson, Karl.: On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. Philosophical Magazine. Series 5 50 (302), pp. 157175 (1900)
[19] Griffiths T L, Steyvers M. Finding scientific topics. Proc. of the National academy of Sciences, 2004, 101(suppl 1): 5228-5235.
[20] S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images," In IEEE Trans. PAMI, 1984, no. 6, pp. 721-741.
[21] Li B, Wang B, Zhou R, et al, "CITPM: A Cluster-Based Iterative Topical Phrase Mining Framework," in DASFAA. Springer, 2016, pp. 197-213.
[22] Kullback, S, Leibler, R.A., "On information and sufficiency. Annals of Mathematical Statistics," 1951, 22(1):7986.
[23] Rodriguez A, Laio A., "Clustering by fast search and find of density peaks." Science, 2014, 344(6191):1492-1496.
[24] Frantzi K, Ananiadou S, Mima H., "Automatic recognition of multi-word terms:. the c-value/nc-value method," International Journal on Digital Libraries, 2000, 3(2): 115-130.
[25] Witten I H, Paynter G W, Frank E, et al., "KEA: practical automatic keyphrase extraction," in ACM Conference on Digital Libraries, 1999, pp. 254-255.
[26] Fisher, Ronald Aylmer., "The design of experiments," 1937, Oliver And Boyd, Edinburgh, London.
[27] Fisher R A., "The arrangement of field experiments[M]/Breakthroughs in statistics". Springer, New York, NY, 1992: 82-91.
[28] Martin, Nick, and Hermine Maes. "Multivariate analysis," 2008.
[29] Can F, Ozkarahan E A. Concepts and effectiveness of the cover-coefficient-based clustering methodology for text databases. ACM TODS, 1990, 15(4): 483-517.
[30] Can F. Incremental clustering for dynamic information processing[J]. ACM TOIS, 1993, 11(2): 143-164.
[31] Zamir O, Etzioni O, Madani O, et al. Fast and Intuitive Clustering of Web Documents. In KDD. 1997, 97:287-290.
[32] Chang J, Gerrish S, Wang C, et al., "Reading tea leaves: How humans interpret topic models," In NIPS, 2009, pp. 288-296.
[33] McDonald J H. Handbook of biological statistics. Baltimore, MD: Sparky House Publishing, 2009.
[34] VanVoorhis W., Morgan B L. Understanding power and rules of thumb for determining sample sizes. Tutorials in Quantitative Methods for Psychology, 2007, 3(2): 43-50.
[35] Brill E., "A Simple Rule-based Part of Speech Tagger," In ACL ANLP, 2002, pp. 152-155.
[36] McDonald R, Crammer K, Pereira F., "Online large-margin training of dependency parsers," in ACL, 2005, pp. 91-98.
[37] Church K W., "A stochastic parts program and noun phrase parser for unrestricted text," in ACL ANLP, 1988, pp. 136-143.
[38] Shen H, Sarkar A., "Voting between multiple data representations for text chunking," Springer Berlin Heidelberg, 2005.
[39] Sha F, Pereira F., "Shallow parsing with conditional random fields," in NAACL HLT, 2003, pp. 134-141.
[40] Vishwanathan N., Schraudolph N., Schmidt W., et al., "Accelerated training of conditional random fields with stochastic gradient methods," in ICML, 2006, pp.969-976.
[41] Wallach, H. M., "Topic modeling: beyond bag-of-words," In ICML, 2006, pp. 977-984.
[42] Blei, David M. and Lafferty, John D., "Visualizing topics with multi-word expressions," Statistics - Machine Learning, 2009.

Bing Li received the master's degree in computer science from Northeastern University, China, in 2013. He is currently working toward the PhD degree at Northeastern University, China. His research interests include data quality, data mining.

Xiaochun Yang received the PhD degree in computer science from Northeastern University, China, in 2001. She is a professor in the Department of Computer Science at Northeastern University, China. Her research interests include data quality, and data privacy. She is a member of the ACM, the IEEE Computer Society, and a senior member of the CCF.



Rui Zhou received the BS and MS degrees from Northeastern University, China in 2004 and 2006. He received the PhD degree from Swinburne University of Technology, Australia in 2010. He is currently a Lecturer at Swinburne University of Technology, Australia. His research interests include data science, algorithms and service computing.

Bin Wang received the PhD degree in computer science from Northeastern University in 2008. He is currently an associate professor in the Computer System Institute at Northeastern University. His research interests include design and analysis of algorithms, queries processing over streaming data, and distributed systems. He is a member of the CCF.

Chengfei Liu received his PhD degree in computer science from Nanjing University, China, in 1988. Currently, he is a professor in the Swinburne University of Technology, Australia. His research interests include keywords search on structured data, query processing and refinement for advanced database applications, query processing on uncertain data and big data, and data-centric workflows. He is a member of the IEEE and the ACM.

Yanchun Zhang received the PhD degree in computer science from The University of Queensland, Australia, in 1991. He is a professor and the director of the Centre for Applied Informatics at Victoria University, Melbourne, Australia. He has been active in areas of database and information systems, Web data management and mining and health information sciences. He was a member of Australian Research Council College of Experts (2008-2010).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2823758, IEEE Transactions on Knowledge and Data Engineering

APPENDIX A

A.1 Proof of Lemma 2

According to Definition 4, any longer phrase is formed by hierarchically merging two sub-phrases. Thus, any l-length phrase $Pr_i = d[i, i+l-1]$ is formed by $l-1$ merge operations, and there is one and only one merge operation occurred on each position s within Pr_i ($s \in [i, i+l-2]$). Therefore, given s is the boundary position of two adjacent chunks, the two chunks need to be merged (i.e., there are possible phrases that could across s) if and only if $\{\bigvee_{i\in I}\bigvee_{j\in J} \upsilon(d[i,s-1],d[s,j])\}$, where $I=\{i \mid d[i,s-1]$ is a phrase$\}$, $J=\{j \mid d[s,j]$ is a phrase$\}$. Thus lemma 2 holds.

A.2 Proof of Theorem 1

The quantity of chunks' boundaries (the position between two adjacent chunks) is $n_s = \lceil \frac{|d|}{q} - 1 \rceil$, and each boundary needs a top-down search whose time complexity is $O(q^3)$. The probability of a phrase being cut by a boundary is $p_c = r \cdot l$, and the number of cut phrases X_p could follow a binomial distribution $X_p \sim Bi(n_s, p_c)$, thus the expected value of cut phrases is $n_s p_c$. In worst case, a cut phrase spans $\lceil \frac{l}{q} \rceil$ chunks, thus it needs $[(1+\lceil \frac{l}{q} \rceil)q]^2$ extra computation cost, and it also leads to a $\lceil \frac{l}{q} \rceil \xi(q)$ filter cost, where $\xi(q)$ could be either $O(q^2)$ or constant depending on the different adopted merge conditions (i.e. exact condition or simple condition). Thus, Theorem 1 holds.

A.3 Proof of Theorem 2

In extreme cases, all those phrases' frequency count into a phrase Pr_i (or not count into Pr_i). Therefore, we have $\min\{\frac{\vartheta'-\epsilon}{1-\epsilon}, \frac{\vartheta'}{1+\epsilon}\} \le \vartheta \le \frac{\vartheta'+\epsilon}{1+\epsilon}$. Thus, the approximation ratio $\eta = \max\{\frac{\vartheta}{\vartheta'}, \frac{\vartheta'}{\vartheta}\} = \max\{1+\epsilon, \frac{1-\rho}{1-\epsilon}\ (0 \le \rho \le 1)\}$. Utilizing the fact that $\varepsilon \le \eta - 1$, we have $\varepsilon(\rho,\epsilon) \le \max\{\epsilon, \frac{\rho-\epsilon}{1-\rho}\}$. Thus Theorem 2 holds.

A.4 Proof of Theorem 3

According to Eq. (1), the optimal segmentation should maximize the following probability:

$$\begin{aligned} p(P^*, S(1,b_m)) &= \prod_{i=1}^{m-1} p(s_i, b_{i+1}|b_i) \cdot p({b_i}^*|s_i, s_{i-1}) \\ &= \begin{array}{l} \prod_{i=1}^{m-2} p(s_i, b_{i+1}|b_i) \cdot p({b_i}^*|s_i, s_{i-1}) \\ \times p(s_{m-1}, b_m|b_{m-1}) \cdot p(b^*_{m-1}|s_{m-1}, s_{m-2}) \end{array} \\ &= \begin{array}{l} p(\{P^* - b_m\}, S(1,b_{m-1})) \\ \times p(b^*_{m-1}|s_{m-1}, s_{m-2}) \cdot p(s_{m-1}, b_m|b_{m-1}). \end{array} \end{aligned}$$

Therefore, the optimal probability is computed based on two parts $p(\{P^* - b_m\}, S(1,b_{m-1})) \cdot p(b^*_{m-1}|s_{m-1}, s_{m-2})$ and $p(s_{m-1}, b_m|b_{m-1})$, where the first part depends on b_{m-1} (for defining the boundary of substring $S(1,b_m)$) and b_{m-2} (i.e., s_{m-2} depends on b_{m-2}). Assuming $MxP[b_{i+1}][b_i]$ is the optimal probability of $p(P^*, S(1,b_{i+1}))$ and b_i is the last but one split position, $MxP[b_{i+1}][b_i]$ can be presented as:

$$MxP[b_{i+1}][b_i] = \max_{b_{i-1}\in\{0,\mathbb{L}\}} \left(\begin{array}{c} MxP[b_i][b_{i-1}] \\ \times p(s_i, b_{i+1}|b_i) \cdot p(b_i^*|s_{i-1}, s_i) \end{array} \right) \quad (14)$$

Eq. (14) is also the recursion formula of Algorithm 2. Thus, Theorem 3 holds.

A.5 Proof of Lemma 3

Algorithm 2 needs to fill an $n * n$ two-dimensional matrix, computing each cell has a $O(n)$ cost, thus the total time complexity is $O(n^3)$.

A.6 Gibbs sampling derivation for CPhrLDA

Here we give the details of the derivation for our CPhrLDA model. Given the joint distribution $P(\mathbf{w}, \mathbf{z}|\boldsymbol{\rho}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\delta})$, we have,

$$\begin{aligned} & P(\mathbf{w}, \mathbf{z}|\boldsymbol{\rho}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\delta}) \\ = \ & P(\boldsymbol{w}|\boldsymbol{z}, \boldsymbol{\rho}, \boldsymbol{\beta}, \boldsymbol{\delta}) \cdot P(\mathbf{z}|\boldsymbol{\alpha}) \\ = \ & \iint P(\boldsymbol{w}|\boldsymbol{z}, \boldsymbol{\rho}, \Phi, \Sigma) P(\Phi|\boldsymbol{\beta}) P(\Sigma|\boldsymbol{\delta}) d\Phi d\Sigma \\ & \times \int P(\boldsymbol{z}|\Theta) P(\Theta|\boldsymbol{\alpha}) d\Theta \\ = \ & \iint \prod_{d=1}^{nd} \prod_{i=1}^{|d|} p(w_i^{(d)}|\phi_z^{(d)}, \sigma_{z_i^{(d)}\rho_j^{(d)}}) \prod_{z=1}^{nt}\prod_{\rho=1}^{np} P(\sigma_{z\rho}|\boldsymbol{\delta}) d\Sigma \\ & \prod_{z=1}^{nt} P(\phi_z|\boldsymbol{\beta}) d\Phi \times \int \prod_{z=1}^{nt} \left(\prod_{i=1}^{|d|} P(z_i^d|\theta_d) P(\theta_d|\boldsymbol{\alpha}) \right) d\Theta \\ = \ & \int \prod_{z=1}^{nt} \left(\prod_{w=1}^{W} \phi_{zw}^{n_w^{(z)}} \frac{\Gamma(\sum_{w=1}^W \beta_w)}{\prod_{w=1}^W \Gamma(\beta_w)} \prod_{w=1}^{W} \phi_{zw}^{\beta_w - 1} \right) d\Phi \\ & \times \int \prod_{z=1}^{nt}\prod_{\rho=1}^{np} \left(\prod_{w=1}^{W} \sigma_{z\rho w}^{n_w^{(z\rho)}} \frac{\Gamma(\sum_{w=1}^W \delta_w)}{\prod_{w=1}^W \Gamma(\delta_w)} \prod_{w=1}^{W} \sigma_{z\rho w}^{\delta_w - 1} \right) d\Sigma \\ & \times \int \prod_{d=1}^{nd} \left(\prod_{z=1}^{nt} \theta_{dz}^{n_d^{(z)}} \frac{\Gamma(\sum_{z=1}^{nt} \alpha_z)}{\prod_{z=1}^{nt} \Gamma(\alpha_z)} \prod_{z=1}^{nt} \theta_{dz}^{\alpha_z - 1} \right) d\Theta \\ \propto \ & \prod_{z=1}^{nt} \frac{\prod_{w=1}^W \Gamma(n_w^{(z)} + \beta_w)}{\Gamma\left(\sum_{w=1}^W (n_w^{(z)} + \beta_w)\right)} \prod_{z=1}^{nt}\prod_{\rho=1}^{np} \frac{\prod_{w=1}^W \Gamma(n_w^{(z\rho)} + \delta_w)}{\Gamma\left(\sum_{w=1}^W (n_z^{z\rho} + \delta_w)\right)} \\ & \prod_{d=1}^{nd} \frac{\prod_{z=1}^{nt} \Gamma(n_d^{(z)} + \alpha_z)}{\Gamma\left(\sum_{z=1}^{nt} (n_d^{(z)} + \alpha_z)\right)} \end{aligned}$$

Utilizing the fact that $\Gamma(x) = (x-1)\Gamma(x-1)$, we can obtain the conditional probability,

$$\begin{aligned} & P(t_j|\mathbf{w}, \mathbf{z}(\neg w_i), \boldsymbol{\rho}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\delta}) \\ \propto \ & (\alpha_{t_j} + n_{t_j} - 1) \cdot \begin{cases} \frac{\beta_{w_i} + n_{w_i}^{t_j} - 1}{\sum_{x=1}^{|d|} (\beta_{w_x} + n_{w_x}^{t_j}) - 1}, & if\ \rho(w_i) = 0, \\ \frac{\delta_{w_i} + n_{w_i}^{t_j \rho(w_i)} - 1}{\sum_{x=1}^{|d|} (\delta_{w_x} + n_{w_x}^{t_j \rho(w_i)}) - 1}, & otherwise, \end{cases} \end{aligned}$$

APPENDIX B

B.1 Case Study

We also provide a case study based on the phrase mining results on 5Conf dataset. Table 11 lists sample phrases that mined from 5Conf dataset. We show some of the four topical phrases and plain phrases (i.e. phrases without topic assignments) . The upper part is the mined phrases by using CQMine, and the lower part is the mined phrases by using TopMine and SegPhrases+. Phrases with different quality and cohesion are prefixed with different abbreviation and fonts (Please see the meaning of different abbreviation below the table).

We first compared topical phrases results that were mined by CQMine and ToPMine. CQMine mined 95% high quality and cohesive phrases, whereas ToPMine only mined 68.3% high quality and cohesive phrases. The left four columns in Table 11 shows some samples of the mined results. All these two approaches mined some high quality and cohesive topical phrases (phrases without prefix or prefixed with *HQC*), and inferior quality phrases (phrases prefixed with IC). Compare with ToPMine, CQMine mined more high quality and cohesive topical phrases. We can see that the phrases with prefix `HQC` have higher quality than those with `IQHC`. For example, `Hierarchical Dirichlet Process` has higher quality than `Hierarchical Dirichlet`. And CQMine mined fewer inferior quality phrases than ToPMine.

Moreover, we showed a sample of plain phrases that were mined by CQMine and SegPhrases+. CQMine mined 96.7% high quality phrases, whereas SegPhrases+ only mined 74.4% high quality phrases. The far right column in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2823758, IEEE Transactions on Knowledge and Data Engineering

TABLE 11
A case study on four topics' phrases mined by CQMine and ToPMine from 5Conf dataset.

CQMine				
TOPIC 1 (AI)	TOPIC 2 (ALGORITHM)	TOPIC 3 (DATABASE)	TOPIC 4 (INFORMATION RETRIEVAL)	*Plain Phrases*
Neural Networks	**(HQC)** Constraint Satisfaction Problem	Database Systems	Information Retrieval	**(HQ)** Content based Image Retrieval
Conditional Random Fields	Dynamic Programming	Relational Databases	Information Extraction	**(HQ)** Information and Knowledge Management
(HQC) Hierarchical Dirichlet Process	Heuristic Search	Active Database	Image Retrieval	**(HQ)** Image Retrieval
Matrix Factorization	**(HQC)** Distributed Constraint Optimization	Large Databases	**(HQC)** Distributed Information Retrieva	**(HQ)** Knowledge Discovery
Markov Random Fields	Genetic Algorithms	**(HQC)** Distributed Database Systems	Text Classification	**(HQ)** Information Retrieval
Topic Models	Belief Propagation	Database Management Systems	Text Categorization	**(HQ)** Frequent Subgraph
Probabilistic Models	Global Optimization	Data Management System	Document Retrieval	Semi supervised Learning
Markov Decision	Lower Bound	Relational Database System	**(HQC)** Information Retrieval System	Keyword Search
Hidden Markov Models	Constraint Propagation	Transaction Processing	XML retrieval	Conditional Random Fields
(HQC) Monte Carlo	Linear Programming	**(IC)** Gaussian Process	Topic Models	Knowledge Representation
Bayesian Networks Learning	Constraint Programming	Information Systems	Text Retrieval	Top k Query
(HQC) Gaussian Mixture Model	Integrity Constraints	Database Applications	Named Entity	Heterogeneous Information Network
Markov Network	Genetic Programming	Deductive Databases	**(IC)** Search Result	**(U)** Dierarchical Dirichlet Process
Latent Dirichlet Allocation	Version Space	**(HQC)** Microsoft SQL	Document Clustering	**(U)** Eigen Vector
(IC) Case Study	Scheduling Problems	Access Control	**(HQC)** Ad Hoc	**(U)** Bit Vector
ToPMine				SegPhrases+
Natural Language	**(IQHC)** Constraint Satisfaction	Database Systems	Information Retrieval	**(IQ)** Content based Image
Topic Models	Computational Complexity	Relational Databases	Information Extraction	**(IQ)** Information and Knowledge
Language Models	Constraint Propagation	Expert System	Image Retrieval	**(IQ)** based Image Retrieval
(IQHC) Hierarchical Dirichlet	**(IQHC)** Distributed Constraint	Recommender Systems	Information Systems	**(IQ)** Knowledge Discovery from
Probabilistic Models	Event Detection	**(IQHC)** Distributed Database	Document Retrieval	**(IQ)** Cross Language Information
Gaussian Process	Materialized Views	Deductive Databases	**(IQHC)** Retrieval System	**(IQ)** Mining Frequent
Graphical Models	Integrity Constraints	Production Systems	Feature Extraction	Semi supervised Learning
(IQHC) Mixture Models	**(IC)** Visual Tracking	**(IC)** Main Memory	Incomplete Information	Keyword Search
(IC) Digital Libraries	Pairwise Constraints	Large Databases	Text Retrieval	Conditional Random Fields
(IC) Pose Estimation	**(IC)** Multiple Views	Database Management Systems	XML Documents	Knowledge Representation
(IC) User Interface	Detection Algorithm	Relational Database System	Information Filtering	Top k Query
Generative Model	Clustering with Constraints	Business Process	Heterogeneous Information	Heterogeneous Information Networks
(IC) Resource Allocation	**(IC)** Publish Subscribe	Transaction Processing	(**IC**) Personal Information	**(U)** What You
(IC) User Behavior	**(IC)** Complex Objects	Spatial Databases	(**IQHC**) Distributed Information	**(U)** based Management
Latent Dirichlet Allocation	Community Detection	**(IC)** XML Query	Information Retrieval System	**(U)** based Methods

HQC: High quality and cohesive topical phrases that only mined by CQMine
IC: Inferior cohesive topical phrases
IQHC: Inferior quality but high cohesive topical phrases
Others: High quality and cohesive topical/plain phrases that mined by all methods
HQ: High quality plain phrases that only mined by CQMine
IQ: Inferior quality plain phrases
U: Unique phrases

Table 11 shows some samples of mined phrases. Both methods could mine some high quality phrases (phrases without prefix or prefixed with *HQ*), inferior quality phrases (phrases with IQ) and also some unique phrases (phrases with prefix **U**). One can observe that, CQMine mined more high quality phrases than SegPhrase+, e.g., `Frequent Subgraph` has higher quality than `Mining Frequent`. Some domain-specific phrases, e.g., `eigen vector` and `bit vector` showed in Table 2 can be extracted by CQMine while they can not be extracted by ToPMine and SegPhrases+. Even though SegPhrases+ mined some unique phrases (e.g., `based Methods`) which can not be mined by CQMine, but obviously they are not quality phrases.