Expert Systems With Applications 171 (2021) 114557

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

High quality error-tolerant phrase mining on text corpus

Jiaying Wang [a,b], Jing Shan [a,b,*], Odafen Ehiaribho Santos [b], Jinling Bao [c]

[a] Shenyang University of Technology, Shenyang, Liaoning, PR China
[b] Shenyang Jianzhu University, Shenyang, Liaoning, PR China
[c] Baicheng Normal University, Baicheng, Jilin, PR China

ARTICLE INFO

Keywords:
Phrase mining
Error tolerant
Scalability
MSC 2010:
00-01
99-00

ABSTRACT

Phrases are widely used in many text-based expert and intelligent systems. Phrase mining is a critical and preprocessing operation for these systems. With the increase of text data, errors in text corpus widely exist. Existing approaches focus on mining phrases on clean text corpus. However, neglecting to handle these errors may generate inaccurate results, which further leads to quality decline. To address the problem, we propose an error-tolerant phrase mining method, which not only conducts phrase mining in text corpus but also correct those phrases from errors. It could help to improve the performance of text-based expert and intelligent systems. To improve the performance and scalability, we propose several efficient and effective techniques to optimize the mining process. Experimental results show that our method achieves higher performance compared with state-of-the-art methods.

1. Introduction

Many expert and intelligent systems (Lopez, Prince, & Roche, 2014; Kazemi, Toral, Way, Monadjemi, & Nematbakhsh, 2017; Gali, Mariescu-Istodor, & Fränti, 2017; Zhu, He, & Zhou, 2019) utilize phrase structures to analyse the semantic meaning of text. Phrase mining (a.k.a. phrase extraction, term recognition) is a process to find phrases from text corpus, in which a phrase is a continuous sequence of words that compose one certain semantic unit in text. It is a fundamental task for text mining (TM), information retrieval (IR) and natural language processing (NLP). As a matter of fact, by transforming the expression of text data from "bag-of-words" to "bag-of-phrases", it helps to expose the deeper and richer semantic concepts of text. Accurate phrase mining, as a preprocessing step of intelligent system, has shown the effectiveness for improving the accuracy of many applications including document classification (Bekkerman & Gavish, 2011), sentiment analysis (Zhang et al., 2014), machine translation (Chiang, 2007), and topic modeling (El-Kishky, Song, Wang, Voss, & Han, 2014).

A lot of methods have already been proposed to solve phrase mining problem, such as rule based methods (Abney, 1991; Smadja, 1993; Dagan & Church, 1994; Daille, 1994; Frantzi, Ananiadou, & Mima, 2000), machine learning based methods (Kudoh & Matsumoto, 2000; Van Halteren, 2000; Carreras & i Villodre, 2003; Shen & Sarkar, 2005; Sun, Morency, Okanohara, Tsuruoka, & Tsujii, 2008) and statistics based methods (Deane, 2005; Parameswaran, Garcia-Molina, & Rajaraman, 2010; El-Kishky et al., 2014; Liu, Shang, Wang, Ren, & Han, 2015; Li, Yang, Wang, & Cui, 2017).

As we know, text data are ubiquitous. Especially, with the explosive growth of information science, mobile internet and social networks, large collections of text data are generated everyday. Among those big text data, errors are quite common. Some of them may be generated by typos and misspellings from people, some others may be caused in the process of optical character recognition (OCR). Besides, many web crawlers and auto data collectors further increase those errors. Since text data are usually semi-structured and unstructured, such errors are not easy to be detected, which brings a big challenge for accurate phrase mining.

The existing methods may generate low quality phrases without considering errors in text data in following reasons. Firstly, if an error in a phrase is not common (i.e. with low frequency), a model without error-tolerance may fail to accept such phrase. For example, consider a scenario in which there is a misspelling of one character repeated in phrase "se**m**antic property" → "se**mm**antic property". We expect that it should be considered to be a phrase. But based on frequency principle (i.e. a phrase should have enough occurrences in the text corpora), existing models cannot take such word sequence as a phrase since there is only one occurrence of such misspelling. Notice that even a stemming method cannot help to accept the result (Madariaga, Castillo, & Hilera, 2005),

* Corresponding author at: Shenyang University of Technology, Shenyang, Liaoning, PR China.
E-mail addresses: jiaying@sjzu.edu.cn (J. Wang), shanjing@sjzu.edu.cn (J. Shan), baojinling@sina.com (J. Bao).

https://doi.org/10.1016/j.eswa.2020.114557
Received 30 March 2019; Received in revised form 14 February 2020; Accepted 30 December 2020
Available online 7 January 2021

since the error happened inside the root instead of the suffix of the word. And such example happens in reality even within a fine collected dataset. Check the example (Filé, Nardiello, & Tirabosco, 1995) in DBLP website by searching typo phrase "semmantic property" (https://dblp.uni-trier.de/search?q = semmantic%20propert).

In addition, even a quite common error can still bring unexpected problems. Because if an error happens with a high frequency, one model may accept both the correct and misspelling text as phrases. However, a succeeding process like document classification or sentiment analysis will take the phrase and its "variant" as different phrases, which may further affect the performance of succeeding process.

Last but not least, some errors can be very difficult to detect without deep analysis of the phrase semantics. For example, consider a scenario that a phrase "part of speech" is misspelled as "part of peach". Since "peach" is also an existing word, it may take "part of peach" instead of "part of speech" as the correct phrase.

The easiest considered way is to adopt an approach with two steps, which first runs an existing text-based data cleaning method (Reynaert, 2014) on text corpora to remove errors, then utilizes an existing phrase mining method to generate phrases. However, such approach still cannot solve the problem ideally. The reason is that detecting and repairing errors in text data itself is a challenging job (Chu, Ilyas, Krishnan, & Wang, 2016). Besides, without considering semantics, we cannot appropriately fix errors. Meanwhile, without clean text data, it is difficult to extract high-quality phrases. So it is a sort of a chicken-or-egg problem. For this consideration, we expect to devise a thorough method to fix errors in the process of phrase mining.

In this paper, we focus on error-tolerant phrase mining method. To the best of our knowledge, this is the first paper to consider error-tolerance in the process of phrase mining.

The main contributions are as follows:

- We propose a novel error-tolerant phrase mining method to extract high quality phrases. The method is based on error-tolerant feature extraction and error-tolerant phrase model.
- Since error-tolerant operation is time consuming, we propose a dynamic programming approach and a trie-based optimization technique to accelerate the mining process.
- We conduct extensive experimental study to demonstrate the efficiency and effectiveness of the proposed method and conduct comparisons with several state-of-the-art methods.

2. Related work

Phrases are widely used in many expert and intelligent systems. Lopez et al. (2014) utilize noun phrases from text to automatically generate titles. Kazemi et al. (2017) utilize phrases to improve the accuracy of machine translations. Gali et al. (2017) utilize phrases to generate titles on web pages. Zhu et al. (2019) use phrases to discover opinions in tweets.

Phrase mining problem for text data has gained a lot of attention in both theory and practical fields. The research about phrase mining starts from NLP community. In 1991, an approach was proposed to parse text into correlated chunks of words based on part-of-speech (POS) tagging technique (Abney, 1991). After that a lot of methods are proposed to extract phrases with rigorous language rules (Smadja, 1993; Dagan & Church, 1994; Daille, 1994; Frantzi et al., 2000).

To overcome high annotation cost and improve precision, several machine learning based methods are proposed (Kudoh & Matsumoto, 2000; Van Halteren, 2000; Carreras & i Villodre, 2003; Shen & Sarkar, 2005; Sun et al., 2008). Those methods take annotated texts as training data, and learn rules based on POS features to recognize phrases in other text data. Based on combining both linguistic knowledge and statistical measures, those methods receive quite good results, but the shortcomings of those methods are poor scalability for large datasets and lack of flexibility to handle different kinds of text corpora.

To improve scalability, several methods (Deane, 2005; Parameswaran et al., 2010; El-Kishky et al., 2014) utilize frequency statistics or its variants in text corpus to do phrase mining. For those methods, neither annotated text as training set nor rigorous language rules are needed. Thus they can process massive corpora efficiently. However, they do not consider errors in the text. Another perspective to do phrase mining is to consider it as a sequence segmentation problem. The approaches in (Liu et al., 2015; Li et al., 2017) take estimated phrase quality as guidance on partition of text sequence, in which phrase quality is defined based on common accepted criteria (i.e. frequency, concordance, completeness, etc.) of a good phrase and computed based on given text data.

Other variant problems include *key-phrase mining problem* and *interesting-phrase mining problem*. Key-phrase mining problem is studied in (Witten, Paynter, Frank, Gutwin, & Nevill-Manning, 1999; Mihalcea & Tarau, 2004; Hasan & Ng, 2010; Liu, Chen, Zheng, & Sun, 2011; Hasan & Ng, 2014). Those studies focus on finding only the most prominent phrases to describe a single document. Interesting-phrase mining problem is studied in (Bedathur, Berberich, Dittrich, Mamoulis, & Weikum, 2010; Gao & Michel, 2012; Deepak, Dey, & Majumdar, 2014). Those studies focus on finding frequently occurring interesting phrases for ad hoc subsets of a corpus.

All aforementioned studies are based on an assumption that there are no errors or only trivial errors in text corpora. However, different from the assumption, errors are common in text. In this paper, we focus on developing a method which is tolerant to errors in text data. The method can still find high-quality phrases on low-quality text.

3. Preliminaries and problem statement

We follow the same way as (Bedathur et al., 2010; El-Kishky et al., 2014) to define a document d, a word w_i and a phrase p_j. A document $d \in D$ can be expressed as $d = w_1 w_2 \ldots w_n$. After the phrase mining, we expect to transform it into a list of phrases $d \Rightarrow p_1 p_2 \ldots p_m$. We utilize s_{p_j} and e_{p_j} to represent the start and end positions of words for phrase p_j, so that we can represent the phrase p_j as a pair of positions $[s_{p_j}, e_{p_j}]$.

We utilize edit distance as a metric to address error-tolerance of phrases. It is the minimum edit operations to transform one string (a.k.a. a sequence of characters) to another. Edit distance is widely adopted to evaluate the dissimilarity of two strings. It can be computed based on a dynamic programming algorithm (Navarro, 2001). We utilize $ed(a, b)$ to represent the edit distance of strings a and b. Further we normalize the result to be a similarity metric to achieve the error tolerance for phrases on character level. We calculate the similarity of two string s_1 and s_2 as depicted in Eq. (1). And it is not difficult to prove that the metric satisfies $0 \leqslant sim(s_1, s_2) \leqslant 1$, and for the same phrase s_1, we have $sim(s_1, s_1) = 1$.

$$sim\left(s_1, s_2\right) = 1 - \frac{ed(s_1, s_2)}{max(|s_1|, |s_2|)} \tag{1}$$

We represent an extracted phrase as $\langle p^*, [s_{p_j}, e_{p_j}] \rangle$, in which we utilize p^* to denote the correct form of phrase. We omit the document d since it is obvious in the context.

Next we utilize a simple example to illustrate the phrase mining problem. Consider an article title "phrase mining from text corpus based on part of speach". Notice that we misspell the "speach" here deliberately. Suppose after phrase mining, the document will be transformed to

a group of disjoint phrases as [phrase mining] [from] [text corpus] [based on] [part of speech]. But the correct result of phrase mining should be $\langle phrasemining, [1,2]\rangle, \langle from, [3,3]\rangle, \langle textcorpus, [4,5]\rangle, \langle basedon, [6,7]\rangle$, and $\langle partofspeech, [8,10]\rangle$. Notice that the phrase containing misspelled word has been corrected. We could further filter the meaningless phrases or stop words in this example, so we generate $\langle phrasemining, [1,2]\rangle, \langle textcorpus, [4,5]\rangle$, and $\langle partofspeech, [8,10]\rangle$ as the final result. Notice that the meaningless part like stop words can still exist in a final result phrase, like the "of" in phrase "part of speech" in this example.

4. Error-tolerant phrase mining method

We first give the framework of our error-tolerant phrase mining method. The whole work consists of four parts. (1) Generate candidate phrases; (2) Extract error-tolerant features from candidate phrases; (3) Construct error-tolerant phrase model based on extracted features; (4) Correct phrases from errors.

4.1. Candidate phrases generation

We utilize a bottom-up approach to generate candidate phrases. We firstly take all words (except the stop words) in text corpus as candidate phrases. Notice that if there is no such stop word dictionary, all words in text corpus are taken as candidate phrases. Then we merge neighbor phrases to generate new candidate phrases. We use C to denote the set of all the current candidate phrases.

Given two neighbor phrases $p_j \in C$ and $p_{j+1} \in C$, we utilize $p_j \widehat{\oplus} p_{j+1}$ to represent a new merged phrase. Notice that the merged phrase may not just concatenate candidate phrases, stop words in the middle of two candidate phrases are also concatenated into the new candidate phrase. In this way, a stop word which plays a joint role in phrase can also be merged into a candidate phrase, although it is not considered to be a candidate phrase initially.

The process of merge operation can be handled efficiently in our framework, since we have start and end pairs $[s_{p_j}, e_{p_j}]$ and $[s_{p_{j+1}}, e_{p_{j+1}}]$ for candidate phrases p_j and p_{j+1}, we can easily generate the merged phrase as $[s_{p_j}, e_{p_{j+1}}]$.

4.2. Error-tolerant feature extraction for candidate phrases

Frequency is an important feature for phrases. We utilize $F(p)$ to represent the frequency of a candidate phrase p in text corpus, which can be calculated directly from document collection. Based on frequency, we can compute the probability of observing a candidate phrase p_j in text corpus as depicted in Eq. (2).

$$P\left(p_j\right) = \frac{F\left(p_j\right)}{\sum_{p\in C} F\left(p\right)} \tag{2}$$

Since phrase may contain errors, which will affect probability statistics. Suppose p_j^* is the correct phrase of p_j. We expect to find $P(p_j^*)$ instead of $P(p_j)$. However, there can exist many similar phrases of p_j and p_j itself has a chance to be the correct phrase, we cannot know which is the correct form of phrase p_j beforehand. To address the problem, we amend phrase probability by considering the similarity of a phrase with any possible similar phrases. We utilize a tangent function to transform the similarity in case to increase the weight of highly similar phrases. The new probability is called error-tolerant probability $\widetilde{P}(p_j)$, which is given in Eq. (3).

$$\widetilde{P}\left(p_j\right) = \sum_{x\in C} \tan\left(\frac{\pi}{\lambda} sim\left(x, p_j\right)\right) \frac{P(x)}{\Sigma_{sim}} + \tan\left(\frac{\pi}{\lambda}\right) \frac{P\left(p_j\right)}{\Sigma_{sim}} \tag{3}$$

Here we utilize Σ_{sim} to denote the summation of the transformed similarity of the phrase p_j with all similar phrases, we have $\Sigma_{sim} = \sum_{x\in C} \tan\left(\frac{\pi}{\lambda} sim\left(x, p_j\right)\right) + \tan\left(\frac{\pi}{\lambda}\right)$. We utilize a threshold of minimal similarity as a constraint. We utilize parameter λ as a coefficient to achieve the balance between probability and similarity, $\lambda \in (2, \pi]$. Notice that we consider phrase p_j itself as the correct phrase in the equation.

Another feature to check whether a phrase is appropriate is based on how much information such phrase holds. We utilize the inverse document frequency (IDF) to evaluate the feature. Similarly we also consider the impact of errors. The error-tolerant IDF is as depicted in Eq. (4).

$$\widetilde{IDF}\left(p_j\right) = \sum_{x\in C} \tan\left(\frac{\pi}{\lambda} sim\left(x, p_j\right)\right) \frac{IDF(x)}{\Sigma_{sim}} + \tan\left(\frac{\pi}{\lambda}\right) \frac{IDF\left(p_j\right)}{\Sigma_{sim}} \tag{4}$$

in which $IDF(p) = log \frac{|D|}{|d\in D; p\in d|}$.

As basic features of candidate phrase p_j, word count and character length of the phrase are also useful to determine whether a phrase is valid. Based on prior knowledge, a document should not contain plenty of very long phrases. To avoid bias on longer candidate phrases, we bring in a length penalty, and define phrase length penalty in Eq. (5).

$$PN\left(\left|p_j\right|\right) = \frac{1}{\left(\left(e_{p_j} - s_{p_j} + 1\right) \times \left(\sum_{i=s_{p_j}}^{e_{p_j}} \left|w_i\right|\right)\right)^{\alpha}} \tag{5}$$

We consider both word count and character length of candidate phrase p_j to compute the penalty. We utilize α as a factor to control the penalty. Notice that we also consider error-tolerance for length penalty. Error may be caused by inserting or deleting characters in the phrase, and we consider inserting and deleting operations have the same probability, so the expectation of character length is still $\left|p_j\right|$.

Another feature needs to be considered is the coherence of candidate phrases. Based on information theory, several metrics can be used to evaluate the coherence of two candidate phrases, such as point-wise Kullback–Leibler divergence (Kullback & Leibler, 1951), point-wise mutual information (Church & Hanks, 1990) and normalized point-wise mutual information (Bouma, 2009). Our approach is based on a variant of normalized point-wise mutual information. We also consider errors in candidate phrases when compute their coherence. Concretely, we could calculate the coherence of a candidate phrase p_j and its succeeding candidate phrase p_{j+1} based on error-tolerant normalized point-wise mutual information, which is defined as Eq. (6). And it satisfies Theorem 1.

$$\widetilde{I_N}\left(p_j, p_{j+1}\right) = \frac{\log\left(\widetilde{P}\left(p_j\right)\widetilde{P}\left(p_{j+1}\right)\right)}{2\log\left(\widetilde{P}\left(p_j \widehat{\oplus} p_{j+1}\right)\right)} \tag{6}$$

Theorem 1. *Error-tolerant normalized point-wise mutual information of two neighbor candidate phrases will always generate a positive value in* $(0, 1]$ *as the result.*

Proof. First consider the formula $\frac{\log((P(x)P(y))}{2\log(P(x,y))}$, in which $P(x)$ and $P(y)$ represent marginal probabilities of random variables X and Y. $P(x, y)$ represents their joint probability.. Since $P(x)P(y) \leqslant 1$ and $P(x, y) \leqslant 1$, we

have $\log((P(x)P(y))\leqslant 0$ and $\log(P(x,\ y))\leqslant 0$, so the whole formula $\frac{\log((P(x)P(y))}{2\log(P(x,y))}\geqslant 0$. We add a minus sign in both the numerator and denominator to make the numerator and denominator positive and the whole value does not change. Now let $P(z) = min(P(x), P(y)), P(x)P(y)\geqslant P(z)^2$ and $P(x,y)\leqslant P(z)$, since it is the joint probability of the two variables. Thus we have $\frac{\log((P(x)P(y))}{2\log(P(x,y))} = \frac{-\log((P(x)P(y))}{-2\log(P(x,y))}\leqslant\frac{-\log(P(z)^2)}{-2\log(P(z))}\leqslant 1$.

Notice that only when two words are mutually exclusive (i.e. $P(x,y) = 0$), we get $\frac{\log((P(x)P(y))}{2\log(P(x,y))} = 0$. We can easily prove that the extreme case cannot happen on two neighbor candidate phrases, since at least they pairwise occur in the document where the two candidate phrases are neighbors. Thus Theorem 1 holds. □

4.3. Error-tolerant phrase model

We solve the phrase mining problem by maximizing global probability of extracted phrases, which is depicted in Eq. (7).

$$P_J\left(C\right) = \frac{\prod_{j=1}^{m}\widetilde{P}\left(p_j\right)\times\widetilde{IDF}\left(p_j\right)\times\widetilde{P}\left(\left|p_j\right|\right)}{\prod_{j=1}^{m-1}\widetilde{I}_N\left(p_j,p_{j+1}\right)} \tag{7}$$

In the equation, we utilize the joint probability of phrase quality (i.e. production of error-tolerant probability, error-tolerant IDF, and length penalty) against the joint probability of error-tolerant normalized point-wise mutual information of those neighbor phrases. In this way, we could maximize the probability of generating error-tolerant high quality phrases meanwhile minimize the probability of failing to find better longer phrases by further merging those candidate phrases. Formally speaking, given a document d, the model will find a group of phrases, that satisfies $P_B = argmax P_J(C)$.

The model works only when the joint probability of phrase quality and the joint probability of error-tolerant normalized point-wise mutual information are larger than 0. It is easy to see that the joint probability of phrase quality is always larger than 0. And based on Theorem 1, we can prove the joint probability of error-tolerant normalized point-wise mutual information is always larger than 0. Notice that the existing coherence metrics (Kullback & Leibler, 1951; Church & Hanks, 1990; Bouma, 2009) do not satisfy this property.

4.4. Phrase correction

After using the error-tolerant phrase model, we can generate a group of candidate phrases and get their start and end positions in the document d. However we cannot directly return candidate phrases as result, since the phrases may contain errors.

We take the phrase correction as a ranking problem, and replace the error phrase with the phrase which has the highest score. Concretely, for a candidate phrase p, we give similar phrase p' a weight based on Eq. (8).

$$\mathscr{W}_{p'} = sim(p',p)\times F(p')\times IDF(p'). \tag{8}$$

The ranking method is similar to term frequency-inverse document frequency (TF-IDF) (Salton & Buckley, 1988), the difference is that we consider similarity of candidate phrases in weight $\mathscr{W}_{p'}$. Notice that it is the correct way to fix errors at the final step of phrase mining, since we have enough knowledge about phrase structure, like the example in Section 3, we will consider the "part of speach" as a whole phrase. If we fix the candidate at the beginning, it may modify the error word "speach" to "peach".

We also utilize an error-tolerant frequency $\widetilde{F}_t$ to further prune those phrases without enough occurrences in the text corpus. This frequency based principle is also adopted in many existing methods (El-Kishky et al., 2014; Liu et al., 2015; Li et al., 2017). Different from those methods, we achieve error-tolerance by merging frequencies of similar phrases.

5. Accelerated mining process

To further accelerate the mining process, we propose a dynamic programming approach to solve the error-tolerant phrase mining problem. We devise an index based on trie structure to perform efficient phrase similarity computation, and also develop improved strategy to accelerate feature extraction based on the index.

5.1. A dynamic programming solution

To solve the error-tolerant phrase mining problem efficiently, we devise a dynamic programming method. Concretely, we utilize a matrix M to store a group of optimal probabilities and a matrix L to store start positions of optimal candidate phrases, in which a cell $M(k,i)$ denotes the best global probability of k phrases from the subset of document $w_1w_2\ldots w_i$, and the start position of the last optimized phase is stored in a cell $L(k,i)$. Initially we set cell $M(1,i)$ and cell $L(1,i)$ as depicted in Eqs. (9) and (10).

$$M\left(1,i\right) = \widetilde{P}\left(d\left[1,i\right]\right)\times\widetilde{IDF}\left(d\left[1,i\right]\right)\times PN\left(\left|d\left[1,i\right]\right|\right) \tag{9}$$

$$L(1,i) = 1 \tag{10}$$

in which $1\leqslant i\leqslant n$. Then we can iteratively compute $P(k,i)$ based on a recursion function as depicted in Eq. (11).

$$M\left(k,i\right) = \max_{j\in[k-1,i-1]} M\left(k-1,j\right) \times\frac{\widetilde{P}\left(d\left[j+1,i\right]\right)\times\widetilde{IDF}\left(d\left[j+1,i\right]\right)\times PN\left(\left|d\left[j+1,i\right]\right|\right)}{\widetilde{I}_N\left(d\left[L\left(k-1,j\right),j\right],d\left[j+1,i\right]\right)} \tag{11}$$

in which $k\in[2,n]$, and $i\in[1,n]$. Then $L(k,i)$ can be computed based on the last candidate phrase p_k which achieves the maximum probability in $M(k,i)$. We compute it as depicted in Eq. (12). Notice that the parameter j denotes the end position of candidate phrase p_{k-1}, we need to add 1 to compute the start position of candidate phrase p_k.

$$L\left(k,i\right) = 1+\operatorname*{argmax}_{j\in[k-1,i-1]} M\left(k-1,j\right) \times\frac{\widetilde{P}\left(d\left[j+1,i\right]\right)\times\widetilde{IDF}\left(d\left[j+1,i\right]\right)\times PN\left(\left|d\left[j+1,i\right]\right|\right)}{\widetilde{I}_N\left(d\left[L\left(k-1,j\right),j\right],d\left[j+1,i\right]\right)} \tag{12}$$

The final maximal probability is $\max_{k=1}^{n} M(k,n)$. And the optimal phrases can be acquired by backtracking the matrix L. Algorithm 1 describes our dynamic programming approach for error-tolerant phrase mining problem. For each document d, we firstly clean the matrices M and L (line 2). Then we compute the first row of the matrices M and L (line 4–5) and iteratively compute other cells in M and L (line 8–9). After generating the two matrices, we can find the maximum probability by search the M matrix (line 10–14). Finally, we backtrack the L matrix to generate the optimal candidate phrases (line 15–20). Notice that since we backtrack the matrix, the first phrase we generated is actually the last one, that is the reason why we reverse the candidate phrases before add them into the final result.

Algorithm 1. GeneratePhrases(D)

Input: D: A corpus containing a collection of documents

Output: $\mathcal{A}$: A set of high quality phrases

```
foreach document d ∈ D do
    /* clean matrices P and L                          */
    M ← ∅, L ← ∅;
    /* initialize cells M[1,i] and L[1,i]              */
    for i ← 1 to |d| do
        compute M(1,i) using Equation 9;
        compute L(1,i) using Equation 10;
    /* compute M[k,i] and L[k, i]                      */
    for k ← 2 to n do
        for i ← 1 to n do
            compute M(k,i) using Equation 11;
            compute L(k,i) using Equation 12;
    Pmax ← 0, Kopt ← 0;
    /* find maximum probability                        */
    for k ← 1 to n do
        if Mmax < M(k,n) then
            Mmax ← M(k,n);
            Kopt ← k;
    /* generates optimal candidate phrases             */
    i ← n, k ← Kopt;
    while i ≠ 1 do
        C.add(d[L(k,i),i]);
        k ← k − 1;
        i ← L(k,i) − 1;
    A.add(reverse(C))
return A;
```

5.2. Trie structure based optimization

We utilize a trie structure to index all candidate phrases. Each candidate phrase is decomposed into characters corresponds to a unique path from the root to a leaf node, which is denoted by a bold circle. And each node on the path has a label of a character in the phrase. Only leaf nodes contain positions of a phrase. Notice that we use "leaf" node to denote it is an end position of a candidate phrase. A leaf node can also have children under it. For each node, we also add other information besides the character, which is computed dynamically. It can help to accelerate the mining process by memorizing that temporary information. Fig. 1 shows an example of the trie structure.

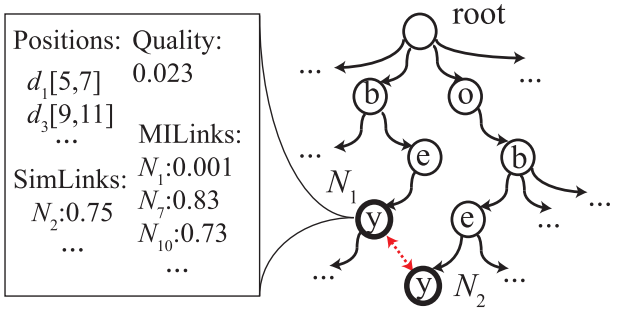

Fig. 1. Trie structure for candidate phrases.

Efficient similarity computation: For each candidate phrase, we need to find its similar phrases. We could efficiently do the similarity comparison using the trie structure. The detailed process of similarity computation using a trie structure can be found in (Ji, Li, Li, & Feng, 2009). The principle is sharing the computation of a group of phrases with the same prefix. In the process of computing optimal phrases, we could compute similar phrases of a candidate phrase many times. To further improve the efficiency of the index, we add a link between a pair of similar leaf nodes, which is called SimLink, so that the next time we visit the node, we could directly get similar phrases with $O(1)$ time complexity. We build SimLinks by memorizing the corresponding leaf node of similar phrases and similarity score.

The similarity threshold θ is an important parameter, since it not only affects the efficiency of error-tolerant operation, but also affects the effectiveness. The method takes longer time to generate candidate phrases for a lower threshold. The value should be set based on error ratio in the corpus. In normal conditions, an error should not happen over 20% of the whole phrases. Thus the threshold θ can be set larger than 80%.

Efficient feature extraction: Similar to the process of storing similarity in SimLinks, we can also memorize the error-tolerant quality for a phrase (denoted by quality) and the mutual information for pair-wise

phrases (denoted by MILinks) in a trie node. In this way, we just need to compute the score once for a unique candidate phrase or candidate phrase pairs. Notice that there can be a large number of candidate phrase pairs, which may need to store a lot of information for a node. We set a threshold τ for the number of MILinks, if its neighbor phrase pair exceed the threshold, we drop those with lower mutual information, since a phrase pair with low value has less probability to occur. In this way, we can achieve a balance between time and space.

6. Experimental evaluation

In this section, we present experimental evaluation of the proposed method. We conducted extensive experiments to study the effectiveness and efficiency. We also compared our method with several state-of-the-art methods.

6.1. Datasets

We evaluated our method on four real datasets.

1. 5Conf[1] contains 44,165 paper titles, which are published in conferences about database, data mining, machine learning, and information retrieval.
2. IMDB[2] contains 98,469 movie reviews from IMDB, which is the world's most popular and authoritative source for movie, TV and celebrity content.
3. Email[3] contains 500,000 emails from employees at Enron corporation. It is acquired by the Federal Energy Regulatory Commission during its investigation after the company's collapse.
4. Amazon[4] contains 1,578,614 customer reviews extracted from Amazon.

6.2. Compared methods

To demonstrate the performance of our approach, we compared with the following state-of-the-art methods:

ToPMine (El-Kishky et al., 2014) is a topical phrase mining method which performs phrase mining and phrase-constrained topic modeling. We used the phrase mining part for comparison.

SegPhrase+ (Liu et al., 2015) is a phrase mining method, which utilizes phrasal segmentation based on rectified frequency, and adds segmentation features to refine quality estimation.

EQPM (Li et al., 2017) is a phrase mining method, which also utilizes phrasal segmentation model to handle overlapping phrases, and utilizes intra-cohesion and inter-isolation in mining phrases.

We also compared with the approach which combines an error correction method with those existing phrase mining methods. The approach contains two steps. The first step is to conduct an error correction method (Reynaert, 2014) as a pre-process to fixed the spelling errors and clean up the text corpus. Then in the second step, we run those state-of-the-art phrase mining methods to generate results. We name the method by combining an error correction method with a phrase mining method, like *EC+ToPMine*. Similarly we name the other two combined methods as *EC+SegPhrase+* and *EC+EQPM*.

6.3. Experimental settings

In our experiments, we firstly studied the parameters (i.e. the balance coefficient λ, the length penalty factor α, and the similarity threshold θ) utilized in our method. We found the generalized best parameters for different datasets.

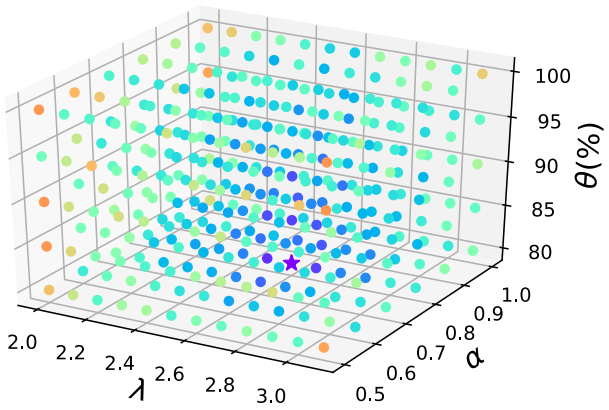

Fig. 2. Parameter selection to find good parameters.

Then we fixed those parameters for all datasets, and conducted comparisons of precision-recall curves with other methods. Concretely, we utilized error-tolerant frequency threshold $\widetilde{F}_t$ to evaluate the effectiveness of our method. For the other methods, we utilized their frequency threshold F_t. Both $\widetilde{F}_t$ and F_t are ranged from 2 to 200.

To further compare the effectiveness in practice, we also conduct experiments by adding 1‰ random errors into the dataset. Concretely, for a certain character in a document, we generate a random float number, and if that number is less than 1‰, we randomly modify the character to another one which is adjacent to it on the keyboard. For example, if s is a character that we choose to modify, it has the same probability to be transformed to a, w, e, d, z or x. In this way, we can achieve a rough simulation of human typos.

Finally we studied the efficiency of our method. We further modified the space threshold τ from 10 to 100 to evaluate the impact on time and space cost.

Our method were implemented in C++ and compiled by G++ version 7.3.0 with "-O3" flags. All the experiments were run on a machine with an 3.50 GHz Intel E3-1240 V5 CPU, 16 GB main memory and 1 TB disk under Ubuntu operating system.

6.4. Evaluation methods

We utilized both objective and subjective evaluation methods to study the effectiveness. They are Wiki phrases based evaluation method and human based evaluation method. Notice that those strategies are also adopted in (Liu et al., 2015; Li et al., 2017) for evaluating the quality of phrase mining.

Wiki Phrases based Evaluation: This evaluation method is an objective method. The experiments were conducted by using Wikipedia phrases as ground truth. Wiki phrases is collected by crawling intra Wiki citations. For fair comparison, only when a generated phrase exactly appears in Wiki phrases, we take the phrase as positive. The precision is defined by the ratio of the number of positive results to the total number. While the recall is defined by the ratio of the number of positive results to the number of all the positive results generated by all the methods. We also evaluated the result by F1-score, which is the harmonic average of precision and recall (i.e. $F_1 = 2\frac{precision \times recall}{precision + recall}$). Notice that all these scores are biased since an appropriate phrase may not be collected in Wiki phrases, and even a phrase is found in Wiki phrases, it may not be the appropriate one. Although the evaluation method is biased, it still has the irreplaceable advantages. Firstly the method is objective. It can be achieved without manual intervention, so it is cheap. And it can statistically reflect the correct trend of those comparing result. Thus we also adopt this evaluation method.

Human based evaluation: Besides Wiki phrases, we also considered

[1] http://web.engr.illinois.edu/~elkishk2/.
[2] https://www.imdb.com/.
[3] https://www.cs.cmu.edu/~./enron/.
[4] https://www.amazon.com/.

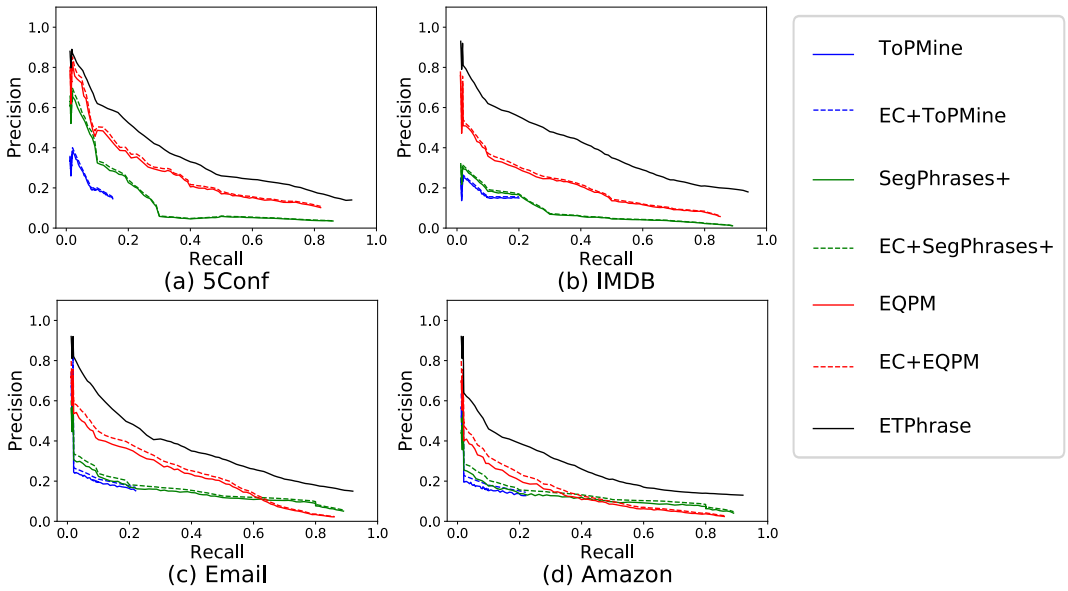

Fig. 3. Comparison of precision-recall curves on different datasets based on Wiki phrases benchmark.

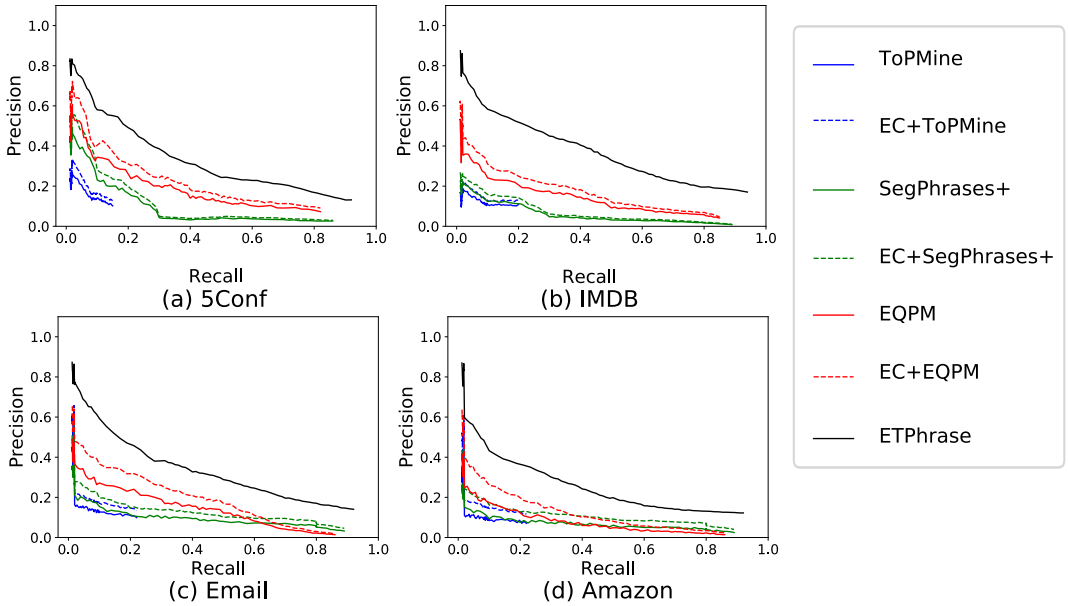

Fig. 4. Comparison of precision-recall curves on different datasets with 1‰ errors based on Wiki phrases benchmark.

to utilize human capability to do the evaluation. This evaluation method is a subjective method. We randomly allocated documents and corresponding generated phrases by different methods to a group of human evaluators (We have invited 87 college students major in computer science as evaluators). We give an evaluator several true or false questions among the results generated by a phrase mining method (e.g. for a sentence "phrase mining from text corpus based on part of speach", a phrase mining method generates following results "{{phrase mining}} from {{text corpus}} based on {{part of speech}}"), which phrases are good, and which are not. Notice that such question can be easily generated based on our phrase representation. Each result of a document was evaluated by 3 evaluators independently. We took the majority opinions as the final result. And the evaluators could utilize a search engine or a dictionary as assistant tools to do judgement.

6.5. *Parameter selection*

The goal of parameter selection is to find a group of good parameters for general text corpus. We focus on three parameters, the balance coefficient λ, the length penalty factor α and the similarity threshold θ. We set the balance coefficient λ from 2.1 to π, the length penalty factor α from 0.5 to 1, the similarity threshold θ from 80% to 100%.

We created a group of models based on different parameters. Considering the efficiency to do such selection, we first made statistics on all datasets, then we did phrase mining on 5% randomly sampled documents in corresponding dataset. To decide which model achieves the best result, we evaluated F1-score based on Wiki phrases, and conducted a summation of every F1-score of each dataset as a final F1-score.

The experimental result is shown in Fig. 2. In the figure, each node represents the result of a model with specified parameters. We utilized different colors to demonstrate different values. The deeper the color, the larger the value. We marked a star on the model with the best score.

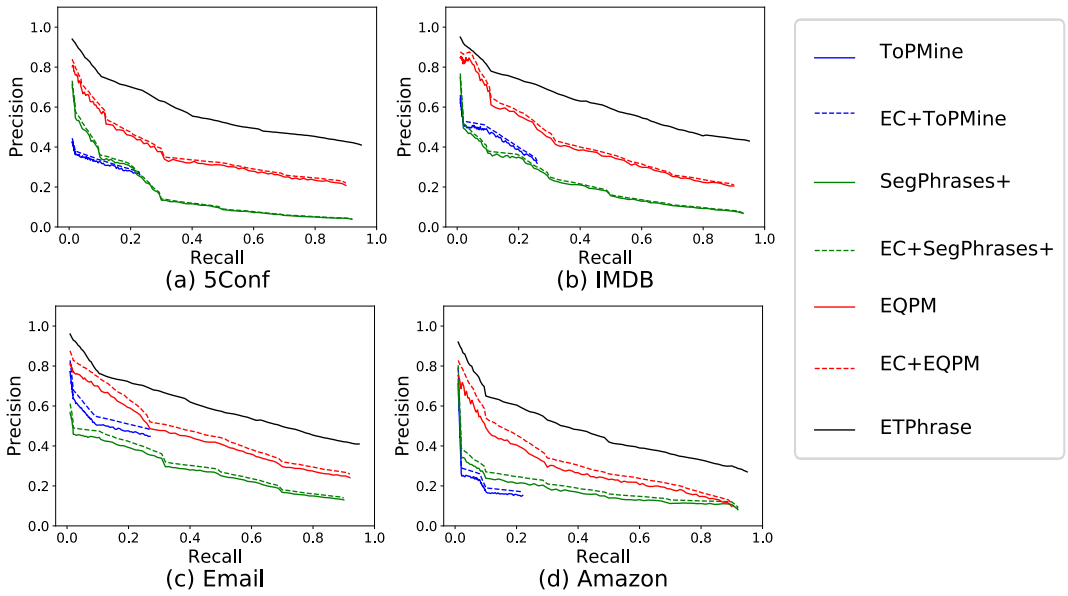

Fig. 5. Comparison of precision-recall curves on different datasets based on human evaluation.

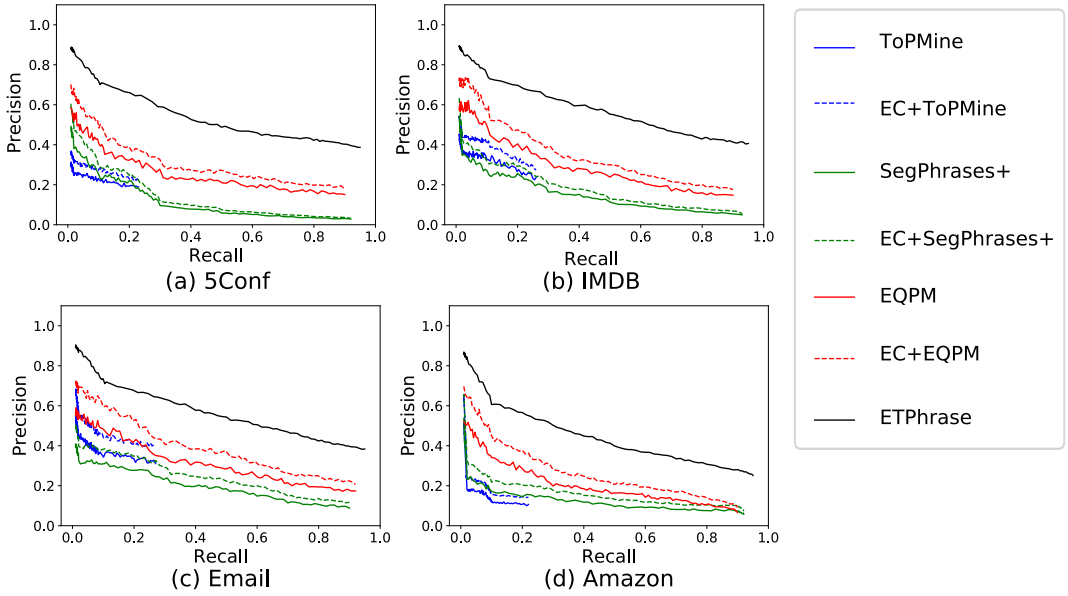

Fig. 6. Comparison of precision-recall curves on different datasets with 1‰ errors based on human evaluation.

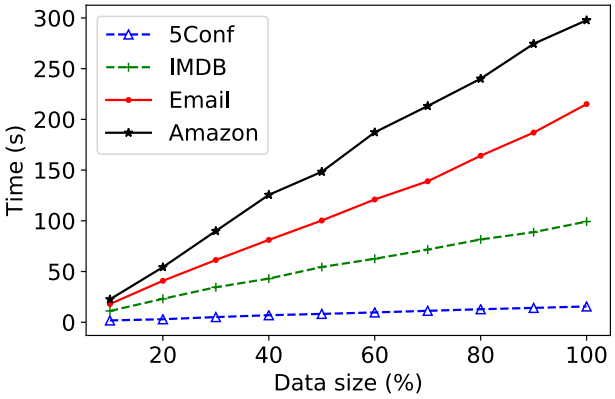

Fig. 7. Scalability of error-tolerant phrase mining.

Table 1
Impact of space threshold τ.

Datasets	$\tau = 10$	$\tau = 40$	$\tau = 70$	$\tau = 100$
5Conf	27.68s	24.55s	21.12s	15.56s
	181.5MB	292.2MB	384.0MB	460.4MB
IMDB	193.86s	179.10s	119.44s	99.24s
	1382.21MB	2473MB	3721MB	3832MB
Email	405.65s	327.17s	318.32s	215.08s
	2712MB	4173MB	6213MB	6832MB
Amazon	566.16s	508.86s	408.51s	297.75s
	3014MB	5254MB	7015MB	7938MB

The best performance was achieved when the balance coefficient λ was 2.6, the length penalty factor α was 0.8, and the similarity threshold θ was 85%.

The impacts of the parameters all have similar trends: with the increase of a parameter, the F1-score first increased, then decreased. The reason is that the parameters impact the precision and recall in opposite way, thus we could find a compromise point.

6.6. Effectiveness evaluation

The goal of effectiveness evaluation is to study how well our method performs on precision and recall. We utilized precision-recall curves to present the result.

Precision-recall curves of different methods evaluated by Wiki phrases are shown in Fig. 3. The trends of precision-recall curves on all datasets are similar. Our proposed method ETPhrase outperforms all the others significantly. With the help of error-tolerance, it achieved much higher recall for the same precision, and achieved much higher precision for the same recall. For different datasets with 1‰ random errors, the precision-recall curves evaluated by Wiki phrases are shown in Fig. 4. We can find the errors further widen the gap between our method and other methods. Although with the help of an existing error correction method, other methods can also achieve the improvement of phrase mining, we can find the improvement is quite limited. The result is consistent with our analysis, that is because without considering semantics, we cannot appropriately fix the errors in the text data. An interesting observation is that we can find our method not only dramatically achieves the improvement in the perspective of precision and recall, but also increases the stability of the precision-recall curves.

The precision-recall curves of different methods evaluated by human evaluators are shown in Fig. 5. Compared with the evaluation by Wiki phrases, all the methods achieved better results. We can find the trends of the curves are consistent. It confirms our analysis that the scores evaluated based on Wiki phrases are biased, but still we can utilize it as a rough measurement to reflect the correct trend of those comparing results. For the different datasets with 1‰ random errors, the precision-recall curves evaluated by human evaluators are shown in Fig. 6. Compared with other methods, our method achieved much better result, which further shows the effectiveness of our method in practice.

6.7. Efficiency evaluation

The goal of efficiency evaluation is to study the scalability of methods. The results are shown in Fig. 7. The most expensive parts of execution time are the string similarity computation and the error-tolerant phrase model, however based on dynamic programming solution and several optimization techniques, we can achieve a nearly linear scalability. It proves that the proposed method has good scalability for phrase mining on large text corpus.

We also studied the impact of space threshold τ by modifying it from 10 to 100. Table 1 shows the results. Based on the threshold we can achieve the tradeoff between time and space. It further shows that our method can be practicable even for environments with limited memory space.

6.8. Discussion

One of the potential strengths of our method is that it considers errors in the text data as common phenomena and solve the problem in the process of phrase mining. Errors of text data are common, which is proved in the experiments, and all approaches can be improved by bringing in the error-tolerant mechanism. And we can find even 1‰ errors may have a big influence on the precision of those approaches.

Although bringing in a spelling correction method as a processing step can help to improve the qualities of existing methods, the better opportunity to fix syntactic errors is in the process of phrase mining. That is because the structure of phrases are important for detecting errors effectively, and errors in text can affect discovery of phrase structure. It is consistent with our discussion of chicken-or-egg problem.

Another important merit of our method is the good scalability. Although considering error tolerance will increase the computation cost, our proposed dynamic programming approach and trie structure based optimization can effectively accelerate our method and achieve near linear performance.

7. Conclusion and future work

In this paper, we introduced a novel approach of error-tolerant phrase mining from text corpus. The main idea is to propose an error-tolerant phrase model to consider errors in the process of phrase mining. We consider edit distance as the similarity metric to achieve character level error-tolerance. The experimental study shows our method achieves satisfying performance.

For future work, we consider several avenues of researches worth investigation. (i) Different similarity metrics such as Cosine, Jaccard and Dice similarities. And the comparison of the approaches using different metrics and (ii) Extend the method to support other languages such as Chinese, Arabic and Japanese. Phrases widely exist in different languages, we will study universal method to extract phrases in different languages. (iii) Smart parameter selection. Currently we conduct experiments to detect the compromise point of precision and recall. A future work is to study automatic parameter selection. (iv) Further improvement of the accuracy. Based on the precision-recall curves, there is still a big space to further improve the accuracy of phrase mining.

CRediT authorship contribution statement

Jiaying Wang: Conceptualization, Methodology, Software, Writing - original draft. **Jing Shan:** Data curation, Writing - review & editing, Supervision. **Odafen Ehiaribho Santos:** Data curation, Visualization, Software, Validation. **Jinling Bao:** Resources, Investigation.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The work was partially supported by the National Natural Science Foundation of China (Nos. 61702346, 61702345), and Youth Seedling Foundation of Liaoning Province (No. lnqn202015). Jing Shan is the corresponding author of this work.

References

Abney, S. P. (1991). Parsing by chunks. In *Principle-based parsing* (pp. 257–278). Springer.

Bedathur, S. J., Berberich, K., Dittrich, J., Mamoulis, N., & Weikum, G. (2010). Interesting-phrase mining for ad-hoc text analytics. *PVLDB, 3*, 1348–1357.

Bekkerman, R., & Gavish, M. (2011). High-precision phrase-based document classification on a modern scale. In *KDD*.

Bouma, G. (2009). Normalized (pointwise) mutual information in collocation extraction. In *Proceedings of GSCL* (pp. 31–40).

Carreras, X., & i Villodre, L. M. (2003). Phrase recognition by filtering and ranking with perceptrons. In RANLP.

Chiang, D. (2007). Hierarchical phrase-based translation. *Computational Linguistics, 33*, 201–228.

Chu, X., Ilyas, I. F., Krishnan, S., & Wang, J. (2016). Data cleaning: Overview and emerging challenges. In *SIGMOD conference*.

Church, K. W., & Hanks, P. (1990). Word association norms, mutual information, and lexicography. *Computational Linguistics, 16*, 22–29.

Dagan, I., & Church, K. W. (1994). Termight: Identifying and translating technical terminology. In *4th Applied natural language processing conference, ANLP 1994, Stuttgart, Germany, October 13-15, 1994* (pp. 34–40).

Daille, B. (1994). Study and implementation of combined techniques for automatic extraction of terminology. The balancing act: Combining symbolic and statistical approaches to language.
Deane, P. (2005). A nonparametric method for extraction of candidate phrasal terms. In *ACL*.
Deepak, P., Dey, A., & Majumdar, D. (2014). Fast mining of interesting phrases from subsets of text corpora. In *EDBT*.
El-Kishky, A., Song, Y., Wang, C., Voss, C. R., & Han, J. (2014). Scalable topical phrase mining from text corpora. *Proceedings of the VLDB Endowment, 8*, 305–316.
Filé, G., Nardiello, G., & Tirabosco, A. (1995). Semmantic properties of CHIP (FD). In *Constraint processing, selected papers* (pp. 225–245).
Frantzi, K. T., Ananiadou, S., & Mima, H. (2000). Automatic recognition of multi-word terms: The c-value/nc-value method. *International Journal on Digital Libraries, 3*, 115–130.
Gali, N., Mariescu-Istodor, R., & Fränti, P. (2017). Using linguistic features to automatically extract web page title. *Expert Systems with Applications, 79*, 296–312.
Gao, C., & Michel, S. (2012). Top-k interesting phrase mining in ad-hoc collections using sequence pattern indexing. In *EDBT*.
Hasan, K. S., & Ng, V. (2010). Conundrums in unsupervised keyphrase extraction: Making sense of the state-of-the-art. In *COLING*.
Hasan, K. S., & Ng, V. (2014). Automatic keyphrase extraction: A survey of the state of the art. In *ACL*.
Ji, S., Li, G., Li, C., & Feng, J. (2009). Efficient interactive fuzzy keyword search. In *WWW*.
Kazemi, A., Toral, A., Way, A., Monadjemi, A., & Nematbakhsh, M. (2017). Syntax- and semantic-based reordering in hierarchical phrase-based statistical machine translation. *Expert Systems With Applications, 84*, 186–199.
Kudoh, T., & Matsumoto, Y. (2000). Use of support vector learning for chunk identification. In *Proceedings of the 2nd workshop on learning language in logic and the 4th conference on computational natural language learning-Vol. 7* (pp. 142–144). Association for Computational Linguistics.
Kullback, S., & Leibler, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics, 22*, 79–86.
Li, B., Yang, X., Wang, B., & Cui, W. (2017). Efficiently mining high quality phrases from texts. In *AAAI* (pp. 3474–3481).
Liu, Z., Chen, X., Zheng, Y., & Sun, M. (2011). Automatic keyphrase extraction by bridging vocabulary gap. In *CoNLL*.
Liu, J., Shang, J., Wang, C., Ren, X., & Han, J. (2015). Mining quality phrases from massive text corpora. *Proceedings ACM-Sigmod international conference on management of data, 2015*, 1729–1744.
Lopez, C., Prince, V., & Roche, M. (2014). How can catchy titles be generated without loss of informativeness? *Expert Systems With Applications, 41*, 1051–1062.
Madariaga, R., Castillo, J., & Hilera, J. R. (2005). A generalization of the method for evaluation of stemming algorithms based on error counting. In *String processing and information retrieval, 12th international conference, SPIRE 2005, Buenos Aires, Argentina, November 2–4, 2005, Proceedings* (pp. 228–233).
Mihalcea, R., & Tarau, P. (2004). Textrank: Bringing order into text. In *EMNLP*.
Navarro, G. (2001). A guided tour to approximate string matching. *ACM Computing Surveys, 33*, 31–88.
Parameswaran, A., Garcia-Molina, H., & Rajaraman, A. (2010). Towards the web of concepts: Extracting concepts from large datasets. *Proceedings of the VLDB Endowment, 3*, 566–577.
Reynaert, M. (2014). Ticclops: Text-induced corpus clean-up as online processing system. In *COLING*.
Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing and Management, 24*, 513–523.
Shen, H., & Sarkar, A. (2005). Voting between multiple data representations for text chunking. In *Canadian conference on AI*.
Smadja, F. (1993). Retrieving collocations from text: Xtract. *Computational Linguistics, 19*, 143–177.
Sun, X., Morency, L., Okanohara, D., Tsuruoka, Y., & Tsujii, J. (2008). Modeling latent-dynamic in shallow parsing: A latent conditional model with improved inference. In *COLING 2008, 22nd international conference on computational linguistics, proceedings of the conference, 18–22 August 2008, Manchester, UK* (pp. 841–848).
Van Halteren, H. (2000). Chunking with wpdv models. In *Proceedings of the 2nd workshop on learning language in logic and the 4th conference on computational natural language learning-Vol. 7* (pp. 154–156). Association for Computational Linguistics.
Witten, I. H., Paynter, G. W., Frank, E., Gutwin, C., & Nevill-Manning, C. G. (1999). Kea: Practical automatic keyphrase extraction. In *ACM DL*.
Zhang, Y., Lai, G., Zhang, M., Zhang, Y., Liu, Y., & Ma, S. (2014). Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *SIGIR*.
Zhu, L., He, Y., & Zhou, D. (2019). Hierarchical viewpoint discovery from tweets using bayesian modelling. *Expert Systems With Applications, 116*, 430–438.