Mining Quality Phrases from Massive Text Corpora

Jialu Liu[†*] Jingbo Shang[†*] Chi Wang[‡] Xiang Ren[†] Jiawei Han[†]
[†]Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, USA
[‡]Microsoft Research, Redmond, WA, USA
[†]{jliu64, shang7, xren7, hanj}@illinois.edu [‡]chiw@microsoft.com

ABSTRACT

Text data are ubiquitous and play an essential role in big data applications. However, text data are mostly unstructured. Transforming unstructured text into structured units (*e.g.*, semantically meaningful phrases) will substantially reduce semantic ambiguity and enhance the power and efficiency at manipulating such data using database technology. Thus mining quality phrases is a critical research problem in the field of databases. In this paper, we propose a new framework that extracts quality phrases from text corpora integrated with phrasal segmentation. The framework requires only limited training but the quality of phrases so generated is close to human judgment. Moreover, the method is scalable: both computation time and required space grow linearly as corpus size increases. Our experiments on large text corpora demonstrate the quality and efficiency of the new method.

1. INTRODUCTION

Mining quality phrases refers to automatically extracting salient phrases from a given corpus. It is a fundamental task for text analytics of various domains, such as science, news, social media and enterprise documents. In these large, dynamic collections of documents, analysts are often interested in variable-length phrases, including scientific concepts, events, organizations, products, slogans and so on. Efficient extraction of quality phrases enable a large body of applications to transform from word granularity to phrase granularity. Examples of such applications include topic tracking [21], OLAP on multi-dimensional text collections [40], and document categorization. For keyword search, the extracted phrases facilitate selective indexing, query suggestion and other tasks. Also, extraction of phrases is critical towards information extraction because many concepts, entities and relations are manifested in phrases.

Though the study of this task originates from the natural language processing (NLP) community, the challenge has been recognized of applying NLP tools in the emerging big data that deviate from rigorous language rules. Query logs, social media messages, and textual transaction records are just a few examples. Therefore, researchers have sought more general data-driven approaches, primarily based on the frequent pattern mining principle [2, 34]. The early work focuses on efficiently retrieving recurring word sequences, but many such sequences do not form meaningful phrases. More recent work filters or ranks them according to frequency-based statistics. However, the raw frequency from the data tends to produce misleading quality assessment, and the outcome is unsatisfactory, as the following example demonstrates.

Example 1 (Raw Frequency-based Phrase Mining). Consider a set of scientific publications and the raw frequency counts of two phrases 'relational database system' and 'support vector machine' and their subsequences in the *frequency* column of Table 1. The numbers are hypothetical but manifest several key observations: (i) the frequency generally decreases with the phrase length; (ii) both good and bad phrases can possess high frequency (*e.g.*, 'support vector' and 'vector machine'); and (iii) the frequency of one sequence (*e.g.*, 'relational database system') and its subsequences can have a similar scale of another sequence (*e.g.*, 'support vector machine') and its counterparts. ■

Obviously, a method that ranks the word sequences solely according to the frequency will output many false phrases such as 'vector machine'. In order to address this problem, different heuristics have been proposed based on comparison of a sequence's frequency and its sub-(or super-)sequences, assuming that a good phrase should have high enough (normalized) frequency compared with its sub-sequences and/or super-sequences [29, 12]. However, such heuristics can hardly differentiate the quality of, *e.g.*, 'support vector' and 'vector machine' because their frequency are so close. Finally, even if the heuristics can indeed draw a line between 'support vector' and 'vector machine' by discriminating their frequency (between 160 and 150), the same separation could fail for another case like 'relational database' and 'database system'.

Using the frequency in Table 1, all heuristics will produce identical predictions for 'relational database' and 'vector machine', guaranteeing one of them wrong. This example suggests the intrinsic limitations of using raw frequency counts, especially in judging whether a sequence is too long (longer than a minimum semantic unit), too short (broken and not informative), or right in length. It is a critical bottleneck for all frequency-based quality assessment.

In this work, we address this bottleneck, proposing to rectify the decisive raw frequency that hinders discovering the true quality of a phrase. The goal of the *rectification* is

*Equal Contribution

SIGMOD'15, May 31–June 4, 2015, Melbourne, Victoria, Australia.

http://dx.doi.org/10.1145/2723372.2751523. .

Table 1: A hypothetical example of word sequence raw frequency

sequence	frequency	phrase?	rectified	sequence	frequency	phrase?	rectified
relational database system	100	yes	70	support vector machine	100	yes	80
relational database	150	yes	40	support vector	160	yes	50
database system	160	yes	35	vector machine	150	no	6
relational	500	N/A	20	support	500	N/A	150
database	1000	N/A	200	vector	1000	N/A	200
system	10000	N/A	1000	machine	1000	N/A	150

to estimate how many times each word sequence should be interpreted in whole as a phrase in its occurrence context. The following example illustrates this idea.

Example 2 (Rectification). Consider the following occurrences of the 6 multi-word sequences listed in Table 1.

1. A ⌈relational database system⌋ for images...
2. ⌈Database system⌋ empowers everyone in your organization...
3. More formally, a ⌈support vector machine⌋ constructs a hyperplane...
4. The ⌈support vector⌋ method is a new general method of ⌈function estimation⌋...
5. A standard ⌈feature vector⌋ ⌈machine learning⌋ setup is used to describe...
6. ⌈Relevance vector machine⌋ has an identical ⌈functional form⌋ to the ⌈support vector machine⌋...
7. The basic goal for ⌈object-oriented relational database⌋ is to ⌈bridge the gap⌋ between... ■

The first 4 instances should provide positive counts to these sequences, while the last three instances should not provide positive counts to 'vector machine' or 'relational database' because they should not be interpreted as a whole phrase (instead, sequences like 'feature vector' and 'relevance vector machine' can). Suppose one can correctly count true occurrences of the sequences, and collect rectified frequency as shown in the *rectified* column of Table 1. The rectified frequency now clearly distinguishes 'vector machine' from the other phrases, since 'vector machine' rarely occurs as a whole phrase.

The success of this approach relies on reasonably accurate rectification. Simple arithmetics of the raw frequency, such as subtracting one sequence's count with its quality super sequence, are prone to error. First, which super sequences are quality phrases is a question itself. Second, it is context-dependent to decide whether a sequence should be deemed a whole phrase. For example, the fifth instance in Example 2 prefers 'feature vector' and 'machine learning' over 'vector machine', even though neither 'feature vector machine' nor 'vector machine learning' is a quality phrase. The context information is lost when we only collect the frequency counts.

In order to recover the true frequency with best effort, we ought to examine the context of every occurrence of each word sequence and decide whether to count it as a phrase. The examination for one occurrence may involve enumeration of alternative possibilities, such as extending the sequence or breaking the sequence, and comparison among them. The test for word sequence occurrences could be expensive, losing the advantage in efficiency of the frequent pattern mining approaches.

Facing the challenge of accuracy and efficiency, we propose a segmentation approach named *phrasal segmentation*, and integrate it with the phrase quality assessment in a unified framework with linear complexity (w.r.t the corpus size). First, the segmentation assigns every word occurrence to only one phrase. In the first instance of Example 2, 'relational database system' are bundled as a single phrase. Therefore, it automatically avoids double counting 'relational database' and 'database system' within this instance. Similarly, the segmentation of the fifth instance contributes to the count of 'feature vector' and 'machine learning' instead of 'feature', 'vector machine' and 'learning'. This strategy condenses the individual tests for each word sequence and reduces the overall complexity while ensures correctness. Second, though there are an exponential number of possible partitions of the documents, we are concerned with those relevant to the phrase extraction task only. Therefore, we can integrate the segmentation with the phrase quality assessment, such that (i) only frequent phrases with reasonable quality are taken into consideration when enumerating partitions; and (ii) the phrase quality guides the segmentation, and the segmentation rectifies the phrase quality estimation. Such an integrated framework benefits from mutual enhancement, and achieves both high quality and high efficiency. Finally, both the phrase quality and the segmentation results are useful from an application point of view. The segmentation results are especially desirable for tasks like document indexing, categorization or retrieval.

The main contributions lie in the following aspects:

- Realizing the limitation of raw frequency-based phrase mining, we propose a segmentation-integrated framework to rectify the raw frequency. To the best of our knowledge, it is the first work to integrate phrase extraction and phrasal segmentation and mutually benefit each other.
- The proposed method is scalable: both computation time and required space grow linearly as corpus size increases. It is easy to parallelize as well.
- Experimental results demonstrate that our method is efficient, generic, and highly accurate. Case studies indicate that the proposed method significantly improves applications like *interesting phrase mining* [5, 17, 28] and *relevant word/phrase search* [27].

2. RELATED WORK

2.1 Quality Phrase Mining

Automatic extraction of quality phrases (*i.e.*, multiword semantic units) from massive, dynamically growing corpora gains increasing attention due to its value in text analytics of various domains. As the origin, the NLP community has conducted extensive studies [38, 16, 30, 41, 4]. With predefined POS rules, one can generate noun phrases from each document[1]. However, such rule-based methods usually suffer

[1] `http://www.nltk.org/howto/chunk.html`

in domain adaptation. Supervised noun phrase chunking techniques [31, 39, 10] leverage annotated documents to automatically learn rules based on POS-tagged corpus. These supervised methods may utilize more sophisticated NLP features such as dependency parser to further enhance the precision [20, 25]. The various kinds of linguistic processing, domain-dependent language rules, and expensive human labeling make it challenging to apply to emerging big corpora.

Another kind of approaches explore frequency statistics in document collections [13, 32, 29, 12, 15]. Most of them leverage a variety of statistical measures derived from a corpus to estimate phrase quality. Therefore, they do not rely on linguistic feature generation, domain-specific rules or large training sets, and can process massive corpora efficiently. In [29], several indicators, including frequency and comparison to super/sub-sequences, were proposed to extract n-grams that are not only popular but also concise as concepts. Deane [13] proposed a heuristic metric over frequency distribution based on Zipfian ranks, to measure lexical association for phrase candidates.

Different from aforementioned methods, this work considers integrating phrasal segmentation with phrase quality estimation to further rectify the inaccurate phrase quality initially estimated, based on local occurrence context. Meanwhile, our phrase quality estimation explores statistical features using rectified phrase frequencies, which provides a principled way to refine the phrase quality assessment.

Our work is also related to *keyphrase extraction* in terms of deriving a variety of statistical measures for finding quality phrases [26, 19, 24]. However, keyphrase extraction focuses on deriving from each single document most prominent phrases, instead of from the entire corpus. In [5, 17, 28], interesting phrases can be queried efficiently for ad-hoc subsets of a corpus, while the phrases are based on simple frequent pattern mining methods.

2.2 Word Sequence Segmentation

In our solution, phrasal segmentation is integrated with phrase quality assessment, as a critical component for rectifying phrase frequency. Formally, phrasal segmentation aims to partition a sequence into disjoint subsequences each mapping to a semantic unit, *i.e.*, word or phrase. In terms of identifying semantic units, existing work includes query segmentation [36, 23], phrase chunking [37, 7, 14], and Chinese word segmentation [35, 9], following either supervised setting on labeled data, or unsupervised setting on large corpus. In [36], Tan and Pang proposed a generative model in unsupervised setting which adopted n-gram frequency from a large corpus and used expectation maximization for computing segment scores. Li *et al.* [23] leveraged query click-through data based on a bigram language model and further refined retrieval model with query segmentation.

The phrasal segmentation step in our proposed framework incorporates the estimated phrase quality as guidance on partition of sequences, which does not rely on external sources such as Wikipedia and click-through data and achieves good efficiency. It mainly serves the purpose of rectifying phrase frequencies.

3. PRELIMINARIES

This paper deals with quality phrase extraction from a large collection of documents. The input documents can be any textual word sequences with arbitrary lengths, such as articles, titles, queries, tags, memos, messages and records. A phrase is a sequence of words that appear contiguously in the text, and serves as a whole (non-composible) semantic unit in certain context of the given documents. Its raw frequency is the total count of its occurrences. There is no universally accepted definition of *phrase quality*. However, it is useful to quantify phrase quality based on certain criteria. We use a value between 0 and 1 to indicate the quality of each phrase, and specify four requirements of a good phrase, which conform with previous work.

- **Popularity:** Since many phrases are invented and adopted by people, it could change over time or occasions whether a sequence of words should be regarded as a non-composible semantic unit. When relational database was first introduced in 1970 [11], 'data base' was a simple composition of two words, and then with its gained popularity people even invented a new word 'database', clearly as a whole semantic unit. 'vector machine' is not a meaningful phrase in machine learning community, but it is a phrase in hardware design. Quality phrases should occur with sufficient frequency in a given document collection.
- **Concordance:** Concordance refers to the collocation of tokens in such frequency that is significantly higher than what is expected due to chance. A commonly-used example of a phraseological-concordance is the two candidate phrases 'strong tea' and 'powerful tea' [18]. One would assume that the two phrases appear in similar frequency, yet in the English language, the phrase 'strong tea' is considered more proper and appears in much higher frequency. Because a concordant phrase's frequency deviates from what is expected, we consider them belonging to a whole semantic unit.
- **Informativeness:** A phrase is informative if it is indicative of a specific topic. 'This paper' is a popular and concordant phrase, but not informative in research publication corpus.
- **Completeness:** Long frequent phrases and their subsets may both satisfy the above criteria. A complete phrase should be interpreted as a whole semantic unit in certain context. In the previous discussion of Example 2, the sequence 'vector machine' does not appear as a complete phrase. Note that a phrase and its subphrase can both be valid in appropriate context. For example, 'relational database system', 'relational database' and 'database system' can all be valid in certain context.

Efficiently and accurately extracting quality phrases is the main goal of this study. For generality, we allow users to provide a few examples of quality phrases and inferior ones. The estimated quality should therefore align with these labeled examples. Previous work has overlooked some of the requirements and made assumptions against them. For example, most work assumes a phrase candidate should either be included as a phrase, or excluded entirely, without analyzing the context it appears. Parameswaran *et al.* [29] assumed that if a phrase candidate with length n is a good phrase, its length $n-1$ prefix and suffix cannot be a good phrase simultaneously. We do not make such assumptions. Instead, we take a context-dependent analysis approach – phrasal segmentation.

A *phrasal segmentation* defines a partition of a sequence into subsequences, such that every subsequence corresponds to either a single word or a phrase. Example 2 shows instances

of such partitions, where all phrases with high quality are marked by brackets $\lceil\rfloor$. The phrasal segmentation is distinct from word, sentence or topic segmentation tasks in natural language processing. It is also different from the syntactic or semantic parsing which relies on grammar to decompose the sentences with rich structures like parse trees. Phrasal segmentation provides the necessary granularity we need to extract quality phrases. The total count of times for a phrase to appear in the segmented corpus is called *rectified frequency*.

It is beneficial to acknowledge that a sequence's segmentation may not be unique, due to two reasons. First, as we mentioned above, a word sequence may be regarded as a phrase or not, depending on the adoption customs. Some phrases, like 'bridge the gap' in the last instance of Example 2, are subject to a user's requirement. Therefore, we seek for segmentation that accommodates the phrase quality, which is learned from user-provided examples. Second, a sequence could be ambiguous and have different interpretations. Nevertheless, in most cases, it does not require perfect segmentation, no matter if such a segmentation exists, to extract quality phrases. In a large document collection, the popularly adopted phrases appear many times in a variety of context. Even with a few mistakes or debatable partitions, a reasonably high quality segmentation (*e.g.*, yielding no partition like 'support $\lceil$vector machine$\rfloor$') would retain sufficient support (*i.e.*, rectified frequency) for these quality phrases, albeit not for false phrases with high raw frequency.

With the above discussions, we have formalizations:

Definition 1 (Phrase Quality). Phrase quality is defined to be the possibility of a multi-word sequence being a coherent semantic unit, according to the above four criteria. Given a phrase v, its phrase quality is:

$$Q(v) = p(\lceil v \rfloor | v) \in [0, 1]$$

where $\lceil v \rfloor$ refers to the event that the words in v compose a phrase. For a single word w, we define $Q(w) = 1$. For phrases, Q is to be learned from data. ∎

For example, a good quality estimator is able to return Q(relational database system) ≈ 1 and Q(vector machine) ≈ 0.

Definition 2 (Phrasal Segmentation). Given a word sequence $C = w_1 w_2 \dots w_n$ of length n, a segmentation $S = s_1 s_2 \dots s_m$ for C is induced by a boundary index sequence $B = \{b_1, b_2, \dots, b_{m+1}\}$ satisfying $1 = b_1 < b_2 < \dots < b_{m+1} = n+1$, where a segment $s_t = w_{b_t} w_{b_t+1} \dots w_{b_t+|s_t|-1}$. Here $|s_t|$ refers to the number of words in segment s_t. Since $b_t + |s_t| = b_{t+1}$, for clearness we use $w_{[b_t, b_{t+1})}$ to denote word sequence $w_{b_t} w_{b_t+1} \cdots w_{b_t+|s_t|-1}$. ∎

Example 3 . Continuing our previous Example 2 and specifically for the first instance, the word sequence and marked segmentation are

C = a relational database system for images

S = / a / relational database system / for / images /

with a boundary index sequence $B = \{1, 2, 5, 6, 7\}$ indicating the location of segmentation symbol /. ∎

Based on these definitions, the main input of quality phrase mining task is a corpus with a small set L of labeled quality phrases and $\bar{L}$ of inferior ones. The corpus can be represented by a giant word sequence $\mathcal{C} = C_1 \dots C_D$, where C_d is the word sequence of document $d, d = 1 \dots D$. Each document can be further partitioned into smaller pieces based on different properties of the corpus, such as sentences according to punctuation. The output is a ranked list of phrases with decreasing quality, together with a segmented corpus.

4. QUALITY PHRASE MINING

We first present the full procedure of phrase mining. Then we introduce each of them in following subsections.

1. Generate frequent phrase candidates according to popularity requirement (Sec. 4.1).
2. Estimate phrase quality based on features about concordance and informativeness requirements (Sec. 4.2).
3. Estimate rectified frequency via phrasal segmentation (Sec. 4.3).
4. Add segmentation-based features derived from rectified frequency into the feature set of phrase quality classifier (Sec. 4.4). Repeat step 2 and 3.
5. Filter phrases with low rectified frequencies to satisfy the completeness requirement as post-processing step.

An complexity analysis for this framework is given at Sec 4.5 to show that both of its computation time and required space grow linearly as the corpus size increases.

4.1 Frequent Phrase Detection

Algorithm 1: Frequent Phrase Detection

Input: Corpus $\mathcal{C}$, minimum support threshold τ.
Output: Raw frequency dictionary f of frequent phrases and words.

```
f ← an empty dictionary
index ← an empty dictionary
for i ← 1 to |C| do
    index[C[i]] ← index[C[i]] ∪ i
while index is not empty do
    index' ← an empty dictionary
    for u ∈ index.keys do
        if |index[u]| ≥ τ then
            f[u] ← |index[u]|
            for j ∈ index[u] do
                u' ← u ⊕ C[j + 1]
                index'[u'] ← index'[u'] ∪ {j + 1}
    index ← index'
return f
```

The task of detecting frequent phrases can be defined as collecting aggregate counts for all phrases in a corpus that satisfy a certain minimum support threshold τ, accordding to the popularity requirement. In practice, one can also set a maximum phrase length ω to restrict the phrase length. Even if no explicit restriction is added, ω is typically a small constant. For efficiently mining these frequent phrases, we draw upon two properties:

1. Downward Closure property: If a phrase is not frequent, then any its super-phrase is guaranteed to be not frequent. Therefore, those longer phrases will be filtered and never expanded.
2. Prefix property: If a phrase is frequent, any its prefix units should be frequent too. In this way, all the frequent phrases can be generated by expanding their prefixes.

The algorithm for detecting frequent phrases is given in Alg. 1. We use $\mathcal{C}[\cdot]$ to index a word in the corpus string and $|\mathcal{C}|$ to denote the corpus size. The $\oplus$ operator is for concatecating two words or phrases. Alg. 1 returns a key-value dictionary f. Its keys are vocabulary $\mathcal{U}$ containing all frequent phrases $\mathcal{P}$, and words $\mathcal{U} \setminus \mathcal{P}$. Its values are their raw frequency.

4.2 Phrase Quality Estimation

Estimating phrase quality from only a few training labels is challenging since a huge number of phrase candidates might be generated from the first step and they are messy. Instead of using one or two statistical measures [16, 30, 15], we choose to compute multiple features for each candidate in $\mathcal{P}$. A classifier is trained on these features to predict quality Q for all unlabeled phrases. For phrases not in $\mathcal{P}$, their quality is simply 0.

We divide the features into two categories according to concordance and informativeness requirements in the following two subsections. Only representative features are introduced for clearness. We then discuss about the classifier in Sec. 4.2.3.

4.2.1 Concordance Features

This set of features is designed to measure concordance among sub-units of a phrase. To make phrases with different lengths comparable, we partition each phrase candidate into two disjoint parts in all possible ways and derive effective features measuring their concordance.

Suppose for each word or phrase $u \in \mathcal{U}$, we have its raw frequency $f[u]$. Its probability $p(u)$ is defined as:

$$p(u) = \frac{f[u]}{\sum_{u' \in \mathcal{U}} f[u']}$$

Given a phrase $v \in \mathcal{P}$, we split it into two most-likely sub-units $\langle u_l, u_r \rangle$ such that *pointwise mutual information* is minimized. Pointwise mutual information quantifies the discrepancy between the probability of their true collocation and the presumed collocation under independence assumption. Mathematically,

$$\langle u_l, u_r \rangle = \arg \min_{u_l \oplus u_r = v} \log \frac{p(v)}{p(u_l)p(u_r)}$$

With $\langle u_l, u_r \rangle$, we directly use the pointwise mutual information as one of the concordance features.

$$PMI(u_l, u_r) = \log \frac{p(v)}{p(u_l)p(u_r)}$$

Another feature is also from information theory, called pointwise Kullback-Leibler divergence:

$$PKL(v \| \langle u_l, u_r \rangle) = p(v) \log \frac{p(v)}{p(u_l)p(u_r)}$$

The additional $p(v)$ is multiplied with pointwise mutual information, leading to less bias towards rare-occurred phrases.

Both features are positively correlated with concordance.

4.2.2 Informativeness Features

Some candidates are unlikely to be informative because they are functional or stopwords. We incorporate the following stopword-based features into the classification process:

- Whether stopwords are located at the beginning or the end of the phrase candidate;

which requires a dictionary of stopwords. Phrases that begin or end with stopwords, such as 'I am', are often functional rather than informative.

A more generic feature is to measure the informativeness based on corpus statistics:

- Average inverse document frequency (IDF) computed over words;

where IDF for a word w is computed as

$$\text{IDF}(w) = \log \frac{|\mathcal{C}|}{|\{d \in [D] : w \in C_d\}|}$$

It is a traditional information retrieval measure of how much information a word provides in order to retrieve a small subset of documents from a corpus. In general, quality phrases are expected to have not too small average IDF.

In addition to word-based features, punctuation is frequently used in text to aid interpretations of specific concept or idea. This information is helpful for our task. Specifically, we adopt the following feature:

- Punctuation: probabilities of a phrase in quotes, brackets or capitalized;

higher probability usually indicates more likely a phrase is informative.

Besides these features, many other signals like knowledge-base entities and part-of-speech tagging can be plugged into the feature set. They are less generic quality estimators and require more training or external resources. It is easy to incorporate these features in our framework when they are available.

4.2.3 Classifier

Our framework can work with arbitrary classifiers that can be effectively trained with small labeled data and output a probabilistic score between 0 and 1. For instance, we can adopt random forest [8] which is efficient to train with a small number of labels. The ratio of positive predictions among all decision trees can be interpreted as a phrase's quality estimation. In experiments we will see that 200–300 labels are enough to train a satisfactory classifier.

Just as we have mentioned, both quality phrases and inferior ones are required as labels for training. To further reduce the labeling effort, some machine learning ideas like PU-learning [22] can be applied to automatically retrieve negative labels. Active learning [33] is another popular mechanism to substantially reduce the number of labels required via iterative training. Moreover, it is possible to train a transferable model on one document collection and adapt it to the target corpus. We plan to explore these directions in future work.

Finally, it's noteworthy that in order to extract features efficiently we have designed an adapted Aho-Corasick Automaton to rapidly locate occurrences of phrases in the corpus. Please refer to the appendix for details about this part.

4.3 Phrasal Segmentation

The discussion in Example 1 points out the limitations of using only raw frequency counts. Instead, we ought to examine the context of every word sequence's occurrence and decide whether to count it as a phrase, as introduced in Example 2. The segmentation directly addresses the completeness requirement, and indirectly helps with the concordance

requirement via rectified frequency. Here we propose an efficient phrasal segmentation method to compute rectified frequency of each phrase. We will see that combined with aforementioned phrase quality estimation, bad phrases with high raw frequency get removed as their rectified frequencies approach zero.

Furthermore, rectified phrase frequencies can be fed back to generate additional features and improve the phrase quality estimation. This will be discussed in the next subsection.

We now propose the phrasal segmentation model integrated with the aforementioned phrase quality Q. Given a word sequence C, and a segmentation $S = s_1 \dots s_m$ induced by boundary index sequence $B = \{b_1, \dots, b_{m+1}\}$, where $s_t = w_{[b_t, b_{t+1})}$, the joint probability is factorized as:

$$p(S, C) = \prod_{t=1}^{m} p\left(b_{t+1}, \lceil w_{[b_t, b_{t+1})} \rfloor \middle| b_t\right) \quad (1)$$

where $p(b_{t+1}, \lceil w_{[b_t, b_{t+1})} \rfloor | b_t)$ is the probability of observing a word sequence $w_{[b_t, b_{t+1})}$ as the t-th quality segment. As segments of a word sequence usually have weak dependence on each other, we assume they are generated one by one for the sake of both efficiency and simplicity.

We now describe the generative model for each segment. Given the start index b_t of a segment s_t, we first generate the end index b_{t+1}, according to a prior distribution $p(|s_t| = b_{t+1} - b_t)$ over phrase lengths. Then we generate the word sequence $w_{[b_t, b_{t+1})}$ according to a multinomial distribution over all segments of length $(b_{t+1} - b_t)$. Finally, we generate an indicator whether $w_{[b_t, b_{t+1})}$ forms a quality segment according to its quality $p(\lceil w_{[b_t, b_{t+1}} \rfloor | w_{[b_t, b_{t+1})}) = Q(w_{[b_t, b_{t+1})})$. We formulate its probabilistic factorization as follows:

$$\begin{aligned} &p(b_{t+1}, \lceil w_{[b_t, b_{t+1})} \rfloor | b_t) = p(b_{t+1}|b_t) p(\lceil w_{[b_t, b_{t+1})} \rfloor | b_t, b_{t+1}) \\ =&p\Big(b_{t+1} - b_t\Big) p\Big(w_{[b_t, b_{t+1})} \Big| |s_t| = b_{t+1} - b_t\Big) Q(w_{[b_t, b_{t+1})}) \end{aligned}$$

The length prior $p(|s_t| = b_{t+1} - b_t)$ is explicitly modeled to counter the bias to longer segments as they result in fewer segments. The particular form of $p(|s_t|)$ we pick is:

$$p(|s_t|) \propto \alpha^{1-|s_t|} \quad (2)$$

Here $\alpha \in R^+$ is a factor called *segment length penalty*. If $\alpha < 1$, phrases with longer length have larger value of $p(|s_t|)$. If $\alpha > 1$, the mass of $p(|s_t|)$ moves towards shorter phrases. Smaller α favors longer phrases and results in fewer segments. Tuning its value turns out to be a trade-off between precision and recall for recognizing quality phrases. At the end of this subsection we will discuss how to estimate its value by reusing labels in Sec. 4.2. It is worth mentioning that such segment length penalty is also discussed by Li *et al.* [23]. Our formulation differs from theirs by posing a weaker penalty on long phrases.

We denote $p(w_{[b_t, b_{t+1})} \Big| |s_t|)$ with $\theta_{w_{[b_t, b_{t+1})}}$ for convenience. For a given corpus $\mathcal{C}$ with D documents, we need to estimate $\theta_u = p(u \Big| |u|)$ for each frequent word and phrase $u \in \mathcal{U}$, and infer segmentation S. We employ the maximum a posteriori principle and maximize the joint probability of the corpus:

$$\sum_{d=1}^{D} \log p(S_d, C_d) = \sum_{d=1}^{D} \sum_{t=1}^{m_d} \log p\left(b_{t+1}^{(d)}, \lceil w_{[b_t, b_{t+1}}^{(d)} \rfloor \middle| b_t^{(d)}\right) \quad (3)$$

Algorithm 2: Dynamic Programming (DP)

Input: Word sequence $C = w_1 w_2 \dots w_n$, phrase quality Q, normalized frequency θ, segment length penalty α.
Output: Optimal segmentation S.
$h_0 \leftarrow 1$, $h_i \leftarrow 0$ $(0 < i \le n)$
denote ω as the maximum phrase length
for $i = 1$ ***to*** n **do**
  for $\delta = 1$ ***to*** ω **do**
    if $h_i \cdot p(b_{t+1} = b_t + \delta, \lceil w_{[i+1, i+\delta+1)} \rfloor \Big| b_t) > h_{i+\delta}$ **then**
      $h_{i+\delta} \leftarrow h_i \cdot p(b_{t+1} = b_t + \delta, \lceil w_{[i+1, i+\delta+1)} \rfloor \Big| b_t)$
      $g_{i+\delta} \leftarrow i$
$i \leftarrow n$
$m \leftarrow 0$
while $i > 0$ **do**
  $m \leftarrow m + 1$
  $s_m \leftarrow w_{g_i+1} w_{g_i+2} \dots w_i$
  $i \leftarrow g_i$
return $S \leftarrow s_m s_{m-1} \dots s_1$

Algorithm 3: Viterbi Training (VT)

Input: Corpus $\mathcal{C}$, phrase quality Q, length penalty α.
Output: θ.
initialize θ with normalized raw frequencies in the corpus
while *not converge* **do**
  $\theta'_u \leftarrow 0, \forall\, u \in \mathcal{U}$
  for $d = 1$ ***to*** D **do**
    $S_d \leftarrow DP(C_d, Q, \theta, \alpha)$ via Alg. 2
    assume $S_d = s_1^{(d)} \cdots s_m^{(d)}$
    for $t = 1$ ***to*** m **do**
      $u \leftarrow w_{[b_t, b_{t+1})}^{(d)}$
      $\theta'_u \leftarrow \theta'_u + 1$
  normalize θ' w.r.t. different length as in Eq. (4)
  $\theta \leftarrow \theta'$
return θ

To find the best segmentation to maximize Eq. (3), one can use efficient dynamic programming (DP) if θ is known. The algorithm is shown in Alg. 2.

To learn θ, we employ an optimization strategy called Viterbi Training (VT) or Hard-EM in the literature [3]. Generally speaking, VT is an efficient and iterative way of parameter learning for probabilistic models with hidden variables. In our case, given corpus $\mathcal{C}$, it searches for a segmentation that maximizes $p(\mathcal{S}, \mathcal{C}|Q, \theta, \alpha)$ followed by coordinate ascent on parameters θ. Such a procedure is iterated until a stationary point has been reached. The corresponding algorithm is given in Alg. 3.

The hard E-step is performed by DP with θ fixed, and the M-step is based on the segmentation obtained from DP. Once the segmentation S is fixed, the closed-form solution of θ_u can be derived as:

$$\theta_u = \frac{\sum_{d=1}^{D} \sum_{t=1}^{m_d} \mathbb{1}_{s_t^{(d)} = u}}{\sum_{d=1}^{D} \sum_{t=1}^{m_d} \mathbb{1}_{|s_t^{(d)}| = |u|}} \quad (4)$$

where $\mathbb{1}$ denotes the identity indicator. We can see that θ_u is the rectified frequency of u normalized by the total

Table 2: Effects of segmentation feedback on phrase quality estimation

phrase	quality before feedback	quality after feedback	problem fixed by feedback
np hard in the strong sense	0.78	0.93	slight underestimate
np hard in the strong	0.70	0.23	overestimate
false positives and false negatives	0.90	0.97	N/A
positives and false negatives	0.87	0.29	overestimate
data base management system	0.60	0.82	underestimate
data stream management system	0.26	0.64	underestimate

Algorithm 4: Penalty Learning

Input: Corpus $\mathcal{C}$, labeled quality phrases L, phrase quality Q, non-segmented ratio r_0.
Output: Desired segment length penalty α.

```
up ← 200, low ← 0
while not converge do
    α ← (up + low)/2
    θ ← VT(C, Q, α) via Alg. 3
    r ← r_0 × |L|
    for i = 1 to |L| do
        S ← DP(L_i, Q, θ, α) via Alg. 2
        if |S| = 1 then
            r ← r − 1
    if r ≥ 0 then
        up ← α
    else
        low ← α
return (up + low)/2
```

frequencies of the segments with length $|u|$. For this reason, we name θ *normalized rectified frequency*.

Note that Soft-EM (*i.e.*, Bawm-Welch algorithm [6]) can also be applied to find a maximum likelihood estimator of θ. Nevertheless, VT is more suitable in our case because:

1. VT uses DP for the segmentation step, which is significantly faster than Bawm-Welch using forward-backward algorithm for the E-step;
2. Majority of the phrases get removed as their θ approaches 0 during iterations, which further speeds up our algorithm.

It has also been reported in [3] that VT converges faster and results in sparser and simpler models for Hidden Markov Model-like tasks. Meanwhile, VT is capable of correctly recovering most of the parameters.

Previously in Eq. (2) we have defined the formula of segment length penalty. There is a hyper-parameter α that needs to be determined outside the VT iterations. An overestimate α will segment quality phrases into shorter parts, while an underestimate of α tends to keep low-quality phrases. Thus an appropriate α reflects the user's trade-off between precision and recall. To judge what α value is reasonable, we propose to reuse the labeled phrases used in the phrase quality estimation. Specifically, we try to search for the maximum value of α such that VT does not segment positive phrases. A parameter r_0 named *non-segmented ratio* controls the trade-off mentioned above. It is the expected ratio of phrases in L not partitioned by dynamic programming. The detailed searching process is described in Alg. 4 where we initially set upper and lower bounds of α and then perform a binary search. In Alg. 4, $|S|$ denotes the number of segments in S and $|L|$ refers to the number of positive labels.

4.4 Feedback as Segmentation Features

Rectified frequencies can help refine the feature generation process in Sec. 4.2 and improve the quality assessment. The motivation behind this feedback idea is explained with the examples shown in Table 2. 'Quality before feedback' listed in the table is computed based on phrase quality estimation introduced in Sec. 4.2. For example, the quality of 'np hard in the strong' is significantly overestimated according to the raw frequency. Once we correctly segment the documents, its frequency will be largely reduced, and we can use it to guide the quality estimator. For another example, The quality of phrases like 'data stream management system' were originally underestimated due to its relatively lower frequency and smaller concordance feature values. Suppose after the segmentation, this phrase is not broken into smaller units in most cases. Then we can feed that information back to the quality estimator and boost the score.

Based on this intuition, we design two new features named *segmentation features* and plug them into the feature set introduced in Sec. 4.2. Given a phrase $v \in \mathcal{P}$, these two segmentation features are defined as:

$$\log \frac{p(S,v)|_{|S|=1}}{\max_{|S|>1} p(S,v)}$$

$$p(S,v)|_{|S|=1} \log \frac{p(S,v)|_{|S|=1}}{\max_{|S|>1} p(S,v)}$$

where $p(S,v)$ is computed by Eq. (1). Instead of splitting a phrase into two parts like the concordance features, we now find the best segmentation with dynamic programming introduced in the phrasal segmentation, which better models the concordance criterion. In addition, normalized rectified frequencies are used to compute these new features. This addresses the context-dependent completeness requirements. As a result, misclassified phrase candidates in the above example can get mostly corrected after retraining the classifier, as shown in Table 2.

A better phrase quality estimator can guide a better segmentation as well. In this way, the loop between the quality estimation and phrasal segmentation is closed and such an integrated framework is expected to leverage mutual enhancement and address all the four phrase quality requirements organically.

Note that we do not need to run quality estimation and phrasal segmentation for many iterations. In our experiments, the benefits brought by rectified frequency can penetrate after the first iteration, leaving performance curves over the next several iterations similar. It will be shown in the appendix.

4.5 Complexity Analysis

Frequent Phrases Detection: Since the operation of Hash table is $\mathcal{O}(1)$, both the time and space complexities are

$\mathcal{O}(\omega|\mathcal{C}|)$. ω is a small constant indicating the maximum phrase length, so this step is linear to the size of corpus $|\mathcal{C}|$.

Feature Extraction: When extracting features, the most challenging problem is how to efficiently locate these phrase candidates in the original corpus, because the original texts are crucial for finding the punctuation and capitalization information. Instead of using some dictionaries to store all the occurrences, we take the advantage of the Aho-Corasick Automaton algorithm and tailor it to find all the occurrences of phrase candidates (some prefix issues are addressed by the adapted version in the appendix). The time complexity is $\mathcal{O}(|\mathcal{C}|+|\mathcal{P}|)$ and space complexity $\mathcal{O}(|\mathcal{P}|)$, where $|\mathcal{P}|$ refers to the total number of frequent phrase candidates. As the length of each candidate is limited by a constant ω, $\mathcal{O}(|\mathcal{P}|) = \mathcal{O}(|\mathcal{C}|)$, so the complexity is $\mathcal{O}(|\mathcal{C}|)$ in both time and space.

Phrase Quality Estimation: As we only labeled a very small set of phrase candidates, as long as the number and depth of decision trees in the random forest are some constant, the training time for the classifier is very small compared to other parts. For the prediction stage, it is proportional to the size of phrase candidates and the dimensions of features. Therefore, it could be $\mathcal{O}(|\mathcal{C}|)$ in both time and space, although the actual magnitude might be smaller.

Viterbi Training: It is easy to observe that Alg. 2 is $\mathcal{O}(n\omega)$, which is linear to the number of words. ω is treated as a constant, and thus the VT process is also $\mathcal{O}(|\mathcal{C}|)$ considering Alg. 3 ususally finishes in a few iterations.

Penalty Learning: Suppose we only require a constant ϵ to check the convergence of the binary search. Then after $log_2 \frac{200}{\epsilon}$ rounds, the algorithm converges. So the number of loops could be treated as a constant. Because VT takes $\mathcal{O}(|\mathcal{C}|)$ time, Penalty learning also takes $\mathcal{O}(|\mathcal{C}|)$ time.

Summary. Because the time and space complexities of all components in our framework are $O(|\mathcal{C}|)$, our proposed framework has a linear time and space complexities and is thus very efficient. Furthermore, the most time consuming parts, including penalty learning and VT, could be easily parallelized because of the nature of independence between documents and sentences.

5. EXPERIMENTAL STUDY

In this section, experiments demonstrate the effectiveness and efficiency of the proposed methods in mining quality phrases and generating accurate segmentation. More complete experiments regarding parameter selection and case studies are reported in the appendix. We begin with the description of datasets.

5.1 Datasets

Two real-world data sets were used in the experiments and detailed statistics are summarized in Table 3.

- The **Academia** dataset[2] is a collection of major computer science journals and proceedings. We use both titles and abstracts in our experiments.
- The **Yelp** dataset[3] provides reviews of 250 businesses. Each individual review is considered as a document.

[2]http://arnetminer.org/citation
[3]https://www.yelp.com/academic_dataset

Table 3: Statistics about datasets

dataset	#docs	#words	#labels
Academia	2.77M	91.6M	300
Yelp	4.75M	145.1M	300

5.2 Compared Methods

To demonstrate the effectiveness of the proposed approach, we compared the following phrase extraction methods.

- **TF-IDF** ranks phrases by the product of their raw frequencies and inverse document frequencies;
- **C-Value** proposes a ranking measure based on frequencies of a phrase used as parts of their super-phrases following a top-down scheme;
- **ConExtr** approaches phrase extraction as a market-baskets problem based on an assumption about relationship between n-gram and prefix/suffix $(n-1)$-gram;
- **KEA**[4] is a supervised keyphrase extraction method for long documents. To apply this method in our setting, we consider the whole corpus as a single document;
- **TopMine**[5] is a topical phrase extraction method. We use its phrase mining module for comparison;
- **ClassPhrase** ranks phrases based on their estimated qualities (removing step 3–5 from our framework);
- **SegPhrase** combines ClassPhrase with phrasal segmentation to filter overestimated phrases based on normalized rectified frequency (removing step 4 from our framework);
- **SegPhrase+** is similar to SegPhrase but adds segmentation features to refine quality estimation. It contains the full procedures presented in Sec. 4.

The first two methods utilize NLP chunking to obtain phrase candidates. We use the JATE[6] implementation of the first two methods, *i.e.*, TF-IDF and C-Value. Both of them rely on OpenNLP[7] as the linguistic processor to detect phrase candidates in the corpus. The rest methods are all based on frequent n-grams and the runtime is dramatically reduced. The last three methods are variations of our proposed method.

It is also worth mentioning that JATE contains several more implemented methods including Weirdness [1]. They are not reported here due to their unsatisfactory performance compared to the baselines listed above.

5.3 Experimental Setting

We set minimum phrase support τ as 30 and maximum phrase length ω as 6, which are two parameters required by all methods. Other parameters required by baselines were set according to the open source tools or the original papers.

For our proposed methods, training labels for phrases were collected by sampling representative phrase candidates from groups of phrases pre-clustered on the normalized feature space by k-means. We labeled research areas, tasks, algorithms and other scientific terms in the Academia dataset as quality phrases. Some examples are 'divide and conquer', 'np complete' and 'relational database'. For the Yelp dataset, restaurants, dishes, cities and other related concepts are

[4]https://code.google.com/p/kea-algorithm
[5]http://web.engr.illinois.edu/~elkishk2/
[6]https://code.google.com/p/jatetoolkit
[7]http://opennlp.apache.org

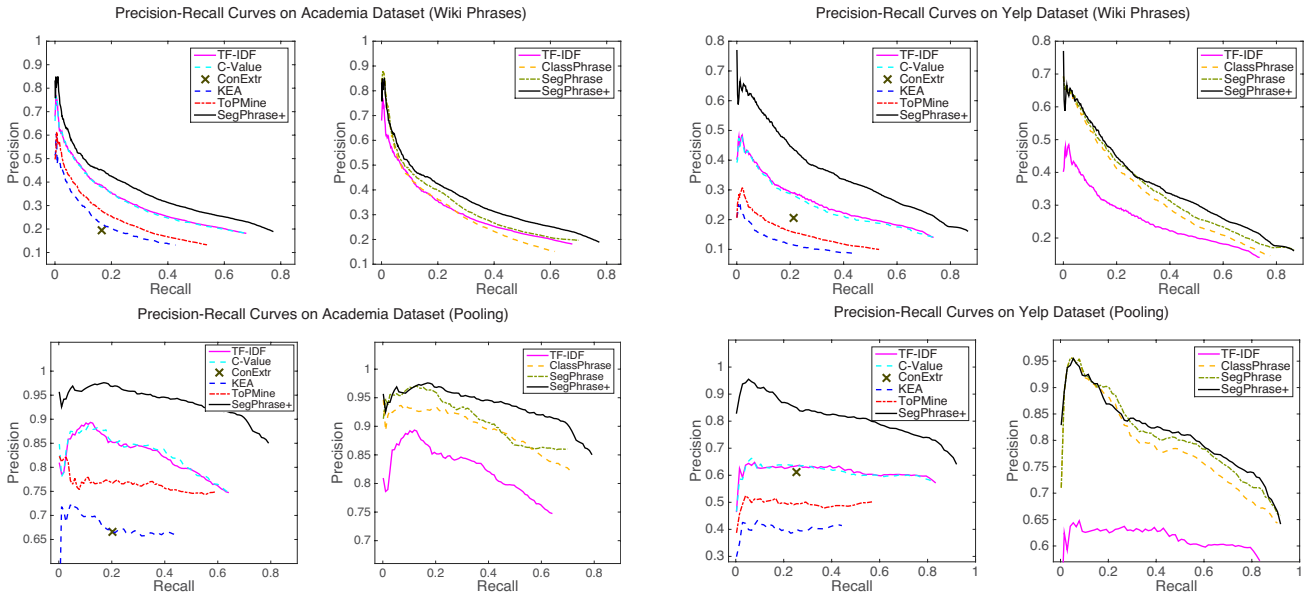

Figure 1: Precision-recall in 4 groups of experiments: (Academia, Yelp) × (Wiki phrase, pooling). Within each group, two figures are shown on par: the left compares SegPhrase+ with existing methods, and the right compares among the 3 proposed methods in this paper together with TF-IDF baseline

labeled to be positive. In contrast, phrases like 'under certain assumptions', 'many restaurants' and 'last night' were labeled as negative. We down-sample low quality phrases because they are dominant over quality phrases. The number of training labels in our experiments are reported in Table 3. To automatically learn the value of segment length penalty, we set the non-segmented ratio r_0 in Alg. 4 as 1.0 for Academia dataset and 0.95 for Yelp dataset. The selection of this parameter will be discussed in detail later in this section.

To make outputs returned by different methods comparable, we converted all the phrase candidates to lower case and merged plural with singular phrases. The phrase lists generated by these methods have different size, and the tail of the lists are low quality. For the simplicity of comparison, we discarded low-ranked phrases based on the minimum size among all phrase lists except ConExtr. ConExtr returns all phrases without ranking. Thus we did not remove its phrases. The remaining size of each list is still reasonably large ($>$ 40,000).

5.4 Evaluation

The goal of our experiments is to study how well our methods perform in terms of "precision" and "recall" and compare with baselines. Precision is defined as the number of quality phrases divided by the number of phrase candidates. Recall is defined as the number of quality phrases divided by the total number of quality phrases.

Wiki Phrases: The first set of experiments were conducted by using Wikipedia phrases as ground truth labels. Wiki phrases refer to popular mentions of entities by crawling intra-Wiki citations within Wiki content. To compute precision, only the Wiki phrases are considered to be positive. For recall, we combine Wiki phrases returned by different methods altogether and view them as all quality phrases. Precision and recall are biased in this case because positive labels are restricted to Wiki phrases. However, we still expect to obtain meaningful insights regarding the performance difference between the proposed and baselines.

Pooling: Besides Wiki phrases, we rely on human evaluators to judge whether the rest of the candidates are good. We randomly sampled k Wiki-uncovered phrases from the returned candidates of each method mentioned at Sec. 5.2. These sampled phrases formed a pool and each of them was then evaluated by 3 reviewers independently. The reviewers could use a popular search engine for the candidates (thus helping reviewers judge the quality of phrases that they were not familiar with). We took the majority of the opinions and used these results to evaluate the methods on how precise the returned quality phrases are. Throughout the experiments we set $k = 500$.

5.5 Results

Precision-recall curves of different methods evaluated by both Wiki phrases and pooling phrases are shown in Fig. 1. The trends on both datasets are similar.

Among the existing work, the chunking-based methods, such as TF-IDF and C-Value, have the best performance; ConExtr reduces to a dot in the figure since its output does not provide the ranking information. Our proposed method, SegPhrase+, outperforms them significantly. More specifically, SegPhrase+ can achieve a higher recall while its precision is maintained at a satisfactory level. That is, many more quality phrases can be found by SegPhrase+ than baselines. Under a given recall, precision of our method is higher in most of the time.

For variant methods within our framework, it is surprising that ClassPhrase could perform competitively to the chunking-based methods like TF-IDF. Note that the latter requires large amounts of pre-training for good phrase chunking. However, ClassPhrase's precision at the tail is slightly worse than TF-IDF on Academia dataset evaluated by Wiki phrases. We also observe a significant difference between

SegPhrase and ClassPhrase, indicating phrasal segmentation plays a crucial role to address the completeness requirement. In fact, SegPhrase already beats ClassPhrase and baselines. Moreover, SegPhrase+ improves the performance of SegPhrase, because of the use of phrasal segmentation results as additional features.

An interesting observation is that the advantage of our method is more significant on the pooling evaluations. The phrases in the pool are not covered by Wiki, indicating that Wikipedia is not a complete source of quality phrases. However, our proposed methods, including SegPhrase+, SegPhrase, and ClassPhrase, can mine out most of them (more than 80%) and keep a very high level of precision, especially on the Academia dataset. Therefore, the evaluation results on the pooling phrases suggest that our methods not only detect the well-known Wiki phrases, but also work properly for the long tail phrases which might occur not so frequently.

From the result on Yelp dataset evaluated by pooling phrases, we notice that SegPhrase+ is a little weaker than SegPhrase at the head. As we know, SegPhrase+ has tried to utilize phrasal segmentation results from SegPhrase to refine the phrase quality estimator. However, segmentation features do not add new information for bigrams. If there are not many quality phrases with more than two words, SegPhrase+ might not have significant improvement and even can perform slightly worse due to the overfitting problem by reusing the same set of labeled phrases. This overfitting problem has been verified in the appendix regarding the iterative SegPhrase+ experiments. In fact, on Academia dataset, the ratios of quality phrases with more than 2 words are 24% among all Wiki phrases and 17% among pooling phrases. In contrast, these statistics go down to to 13% and 10% on Yelp dataset, which verifies our conjecture and explains why SegPhrase+ has slightly lower precision than SegPhrase at the head.

5.6 Model Selection

The goal of model selection is to study how well our methods perform in terms of "precision" and "recall" on various candidate models with different parameters. We specifically want to study two potentially interesting questions:

- How many labels do we need to achieve good results of phrase quality estimation?
- How to choose non-segmented ratio r_0 for deciding segment length penalty?

Experiments for other parameters are provided in the appendix due to the space limit.

5.6.1 Number of Labels

To evaluate the impact of training data size on the phrase quality estimation, we focus on studying the classification performance of ClassPhrase. Table 4 shows the results evaluated among phrases with positive predictions (*i.e.*, $\{v \in \mathcal{P} : Q_v \geq 0.5\}$). With different numbers of labels, we report the precision, recall and F1 score judged by human evaluators (Pooling). The number of correctly predicted Wiki phrases is also provided together with the total number of positive phrases predicted by the classifier. From these results, we observe that the performance of the classifier becomes better as the number of labels increases. Specifically, on both datasets, the recall rises up as the number of labels increases, while the precision goes down. The reason is the down-sampling of low quality phrases in the training data. Overall, the F1 score is monotonically increasing, which indicates that more labels may result in better performance. 300 labels are enough to train a satisfactory classifier.

Table 4: Impact of training data size on ClassPhrase (Top: Academia, Bottom: Yelp)

#labels	prec.	recall	f1	#wiki phr.	#total
50	0.881	0.372	0.523	6,179	24,603
100	0.859	0.430	0.573	6,834	30,234
200	0.856	0.558	0.676	8,196	40,355
300	0.760	0.811	0.785	11,535	95,070

#labels	prec.	recall	f1	#wiki phr.	#total
50	0.948	0.491	0.647	6,985	79,091
100	0.948	0.540	0.688	6,692	57,018
200	0.948	0.554	0.700	6,786	53,613
300	0.944	0.559	0.702	6,777	53,442

5.6.2 Non-segmented Ratio

The non-segmented ratio r_0 is designed for learning segment length penalty, which further controls the precision and recall phrasal segmentation. Empirically, under higher r_0, the segmentation process will favor longer phrases, and vice versa. We show experimental results in Table 5 for models with different values of r_0. The evaluation measures are similar to the previous setting but they are computed based on the results of SegPhrase. One can observe that the precision increases with lower r_0, while the recall decreases. It is because phrases are more likely to be segmented into words by lower r_0. High r_0 is generally preferred because we should preserve most positive phrases in training data. We select $r_0 = 1.00$ and 0.95 for Academia and Yelp datasets respectively, because quality phrases are shorter in Yelp dataset than in Academia dataset.

Table 5: Impact of non-segmented ratio r_0 on SegPhrase (Top: Academia, Bottom: Yelp)

r_0	prec.	recall	f1	#wiki phr.	#total
1.00	0.816	0.756	0.785	10,607	57,668
0.95	0.909	0.625	0.741	9,226	43,554
0.90	0.949	0.457	0.617	7,262	30,550
0.85	0.948	0.422	0.584	7,107	29,826
0.80	0.944	0.364	0.525	6,208	25,374

r_0	prec.	recall	f1	#wiki phr.	#total
1.00	0.606	0.948	0.739	7,155	48,684
0.95	0.631	0.921	0.749	6,916	42,933
0.90	0.673	0.846	0.749	6,467	34,632
0.85	0.714	0.766	0.739	5,947	28,462
0.80	0.725	0.728	0.727	5,729	26,245

5.7 Computational Complexity Study

The following execution time experiments were all conducted on a machine with 20 cores of Intel(R) Xeon(R) CPU E5-2680 v2 @ 2.80GHz. Our framework is mainly implemented in C++ while a small part of preprocessing is in Python. As shown in Fig. 3, the linear curves of total runtime of SegPhrase+ on different proportions of data verifies our linear time complexity analyzed in Sec. 4.5.

dataset	file size	#words	time
Academia	613MB	91.6M	0.595h
Yelp	750MB	145.1M	0.917h

Besides, the pies in Fig. 4 show the ratios of different components of our framework. One can observe that Feature Extraction and Phrasal Segmentation occupy most of the

runtime. Fortunately, almost all components of our frameworks can be parallelized, such as Feature Extraction, Phrasal Segmentation and Quality Estimation, which are the most expensive parts of execution time. It is because sentences can be proceeded one by one without any impact on each other. Therefore, our methods could be very efficient for massive corpus using parallel and distributed techniques. Here we do not compare the runtime with other baselines because they are implemented by different programming languages and some of them further rely on various third-party packages. Among existing implementations, our method is empirically one of the fastest.

5.8 Feature Contribution

We conducted experiments to show the contributions of different features in the classifier of SegPhrase+ in Fig. 2. Feature contributions are determined by aggregating mean squared errors for nodes in decision trees of the random forest classifier. Generally speaking, we can observe that the contributions of features in different aspects vary for two datasets. As the Yelp dataset is noisier and less formal compared to the Academia dataset, the stopword and IDF features are more important in Yelp, whereas the concordance and segmentation features are more important in Academia. Also, due to the subjectivity in Yelp review data, its punctuation feature is less significant. Fig. 2 indicates that supervision is necessary for phrase quality assessment. Meanwhile, it suggests our proposed criteria for judging phrase quality is reasonable since each feature can be a strong signal of the final phrase quality in certain datasets.

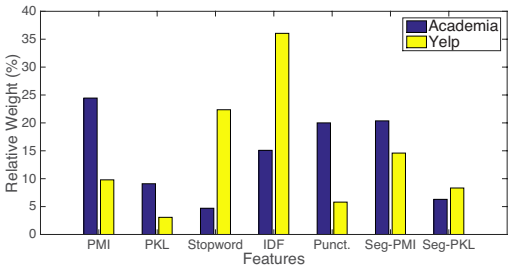

Figure 2: Contributions of features for phrase quality estimation of SegPhrase+

5.9 Case Study

Previous experiments are focused on evaluating phrase quality quantitatively. In this subsection, we show two case studies based on applications taking segmented corpora as input. Recall that the segmented corpus is the other output of the proposed phrase mining framework. In the appendix, we provide a fraction of the ranked phrase list produced by our framework.

5.9.1 *Interesting Phrase Mining*

The first application is to mine interesting phrases in a subset of given corpus. Interesting phrases are defined to be phrases frequent in the subset $\mathcal{C}'$ but relatively infrequent in the overall corpus $\mathcal{C}$ [5, 17, 28]. Given a phrase v, its interestingness is measured by $freq(v, \mathcal{C}') \cdot purity(v, \mathcal{C}', \mathcal{C}) = freq(v, \mathcal{C}')^2 / freq(v, \mathcal{C})$, which considers both phrase frequency and purity in the subset.

We list a fraction of interesting phrases in Table 6 mined from papers published in SIGMOD and SIGKDD conferences. Each series of proceedings form a subset of the whole Academia corpus. Two segmentation methods are compared. The first one relies on dynamic programming using phrase quality estimated by SegPhrase+. The other is based on the phrase chunking method adopted in JATE, which is further used to detect phrase candidates for TF-IDF and C-Value methods. To be fair, we only show phrases extracted by SegPhrase+, TF-IDF and C-Value methods in the table. Because TF-IDF and C-Value perform similarly and they both rely on the chunking method, we merge their phrases and report mining results in one column named 'Chunking'. Phrases in SegPhrase+ but missing in the chunking results are highlighted in purple (red vice versa). One can observe that the interesting phrases mined by SegPhrase+ based on the segmentation result are more meaningful and the improvement is significant. Relatively speaking, phrases mined from the chunking method are of inferior quality. Therefore, many of them are not covered by SegPhrase+. For more experimental results regarding this application, readers can refer to the appendix.

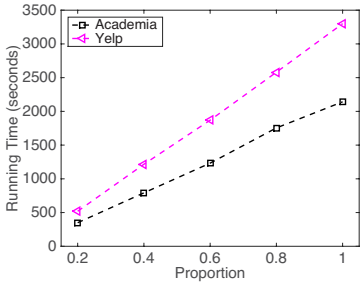

Figure 3: Runtime on different proportions of data

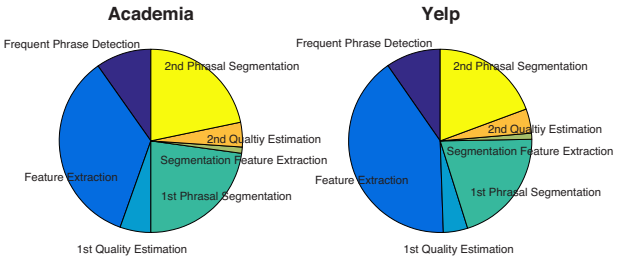

Figure 4: Runtime of different modules in our framework on Academia and Yelp dataset

5.9.2 *Word/Phrase Similarity Search*

With a segmented corpus, one could train a model to learn *distributed vector representations* of words and phrases [27]. Using this technique, words and phrases are mapped into a vector space such that semantically similar words and phrases have similar vector representations. It helps other text mining algorithms to achieve better performance by grouping similar units. The quality of the learned vector representation is closely related to the quality of the input segmented corpus. Accurate segmentation results in good vector representation and this performance gain is usually evaluated by comparing similarity scores between word/phrase pairs. To be specific, one could compute top-k similar words or phrases given a query and compare the ranked lists. We use this to verify the utility of both quality phrase mining and quality segmentation.

Table 6: Interesting phrases mined from papers published in SIGMOD and SIGKDD conferences

Conference	SIGMOD		SIGKDD	
Method	SegPhrase+	Chunking	SegPhrase+	Chunking
1	data base	data base	data mining	data mining
2	database system	database system	data set	association rule
3	relational database	query processing	association rule	knowledge discovery
4	query optimization	query optimization	knowledge discovery	frequent itemset
5	query processing	relational database	time series	decision tree
...	...	...	...	...
51	sql server	database technology	association rule mining	search space
52	relational data	database server	rule set	domain knowledge
53	data structure	large volume	concept drift	important problem
54	join query	performance study	knowledge acquisition	concurrency control
55	web service	web service	gene expression data	conceptual graph
...	...	...	...	...
201	high dimensional data	efficient implementation	web content	optimal solution
202	location based service	sensor network	frequent subgraph	semantic relationship
203	xml schema	large collection	intrusion detection	effective way
204	two phase locking	important issue	categorical attribute	space complexity
205	deep web	frequent itemset	user preference	small set
...	...	...	...	...

Table 7: Top-5 similar words or phrases for representative queries (Top: Academia, Bottom: Yelp)

Query	data mining		olap	
Method	SegPhrase+	Chunking	SegPhrase+	Chunking
1	knowledge discovery	driven methodologies	data warehouse	warehouses
2	text mining	text mining	online analytical processing	clustcube
3	web mining	financial investment	data cube	rolap
4	machine learning	knowledge discovery	olap queries	online analytical processing
5	data mining techniques	building knowledge	multidimensional databases	analytical processing

Query	blu-ray		noodle		valet parking	
Method	SegPhrase+	Chunking	SegPhrase+	Chunking	SegPhrase+	Chunking
1	dvd	new microwave	ramen	noodle soup	valet	huge lot
2	vhs	lifetime warranty	noodle soup	asian noodle	self-parking	private lot
3	cd	recliner	rice noodle	beef noodle	valet service	self-parking
4	new release	battery	egg noodle	stir fry	free valet parking	valet
5	sony	new battery	pasta	fish ball	covered parking	front lot

We show the results in Table 7 from SegPhrase+ and the chunking method mentioned in the previous interesting phrase mining application. Queries were chosen to be capable of showing the difference between the two methods for both Academia and Yelp datasets. Distributed representations were learned through an existing tool [27] and ranking scores were computed based on cosine similarity. Additional comparisons are provided in the appendix.

From the table, one can easily tell that the rank list from SegPhrase+ carries more sense than that from phrase chunking. One of the possible reasons is that chunking method only detects noun phrases in the corpus, providing less accurate information of phrase occurrences than SegPhrase+ to the vector representation learning algorithm.

6. CONCLUSIONS

In this paper, we introduced a novel framework for extracting quality phrases from text corpora. The framework is data-driven and requires only limited training to achieve outstanding performance. The key idea is to rectify the raw frequency of phrases which misleads quality estimation. We develop a segmentation-integrated approach for this purpose, which significantly boosts the final quality of extracted phrases. It is the first work to explore the mutual benefit of phrase extraction and phrasal segmentation. By intergrating them, this work addresses a fundamental limitation of phrase mining and empowers widespread applications. Meanwhile, the method is scalable: both computation time and required space grow linearly as corpus size increases.

A number of open problems need to be solved to allow further development of the proposed phrase mining framework. One direction is to investigate reducing the number of training labels. The current framework requires the model to be trained by 300 labeled phrases. It would be preferable if a transferable model can be pretrained on general document collection and adapt itself to the target corpus. An alternative is to use a domain-independent phrase collection as the training source. Overlapped phrases are first identified and then used to serve as the positive labels. Another direction is to replace Viterbi Training by other parameter estimation approaches to further improve the phrasal segmentation. For instance, one can find top-k best segmentations instead of one for each text snippet. This is less deterministic than the current design but consumes more computational power. Finally, it is now possible to reconsider many text data management problems with this technique. For example, text can be represented by bag of phrases instead of words. The best way to exploit this representation is open for exploration.

7. ACKNOWLEDGEMENTS

Research was sponsored in part by the U.S. Army Research Lab. under Cooperative Agreement No. W911NF-09-2-0053 (NSCTA), the Army Research Office under Cooperative Agreement No. W911NF-13-1-0193, National Science Foundation IIS-1017362, IIS-1320617, and IIS-1354329, HDTRA1-10-1-0120, trans-NIH Big Data to Knowledge (BD2K) initiative grant 1U54GM114838 awarded by NIGMS, and MIAS, a DHS-IDS Center for Multimodal Information Access and Synthesis at UIUC.

References

[1] K. Ahmad, L. Gillam, and L. Tostevin. University of surrey participation in trec8: Weirdness indexing for logical document extrapolation and retrieval (wilder). In *The Eighth Text REtrieval Conference (TREC-8)*, Gaithersburg, Maryland.

[2] H. Ahonen. Knowledge discovery in documents by extracting frequent word sequences. *Library Trends*, 48(1), 1999.

[3] A. Allahverdyan and A. Galstyan. Comparative analysis of viterbi training and maximum likelihood estimation for hmms. In *NIPS*, pages 1674–1682, 2011.

[4] T. Baldwin and S. N. Kim. Multiword expressions. *Handbook of Natural Language Processing, second edition. Morgan and Claypool*, 2010.

[5] S. Bedathur, K. Berberich, J. Dittrich, N. Mamoulis, and G. Weikum. Interesting-phrase mining for ad-hoc text analytics. *VLDB*, 3(1-2):1348–1357, 2010.

[6] C. M. Bishop. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.

[7] G. Blackwood, A. De Gispert, and W. Byrne. Phrasal segmentation models for statistical machine translation. In *COLING*, 2008.

[8] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[9] P.-C. Chang, M. Galley, and C. D. Manning. Optimizing chinese word segmentation for machine translation performance. In *ACL Workshop on Statistical Machine Translation*, 2008.

[10] K.-h. Chen and H.-H. Chen. Extracting noun phrases from large-scale texts: A hybrid approach and its automatic evaluation. In *ACL*, 1994.

[11] E. F. Codd. A Relational Model for Large Shared Data Banks. *Communications of The ACM*, 13:377–387, 1970.

[12] M. Danilevsky, C. Wang, N. Desai, X. Ren, J. Guo, and J. Han. Automatic construction and ranking of topical keyphrases on collections of short documents. In *SDM*, 2014.

[13] P. Deane. A nonparametric method for extraction of candidate phrasal terms. In *ACL*, 2005.

[14] H. Echizen-ya and K. Araki. Automatic evaluation method for machine translation using noun-phrase chunking. In *ACL*, 2010.

[15] A. El-Kishky, Y. Song, C. Wang, C. R. Voss, and J. Han. Scalable topical phrase mining from text corpora. *VLDB*, 8(3), Aug. 2015.

[16] K. Frantzi, S. Ananiadou, and H. Mima. Automatic recognition of multi-word terms:. the c-value/nc-value method. *JODL*, 3(2):115–130, 2000.

[17] C. Gao and S. Michel. Top-k interesting phrase mining in ad-hoc collections using sequence pattern indexing. In *EDBT*, 2012.

[18] M. A. Halliday. Lexis as a linguistic level. *In memory of JR Firth*, pages 148–162, 1966.

[19] K. S. Hasan and V. Ng. Conundrums in unsupervised keyphrase extraction: making sense of the state-of-the-art. In *COLING*, 2010.

[20] T. Koo, X. Carreras, and M. Collins. Simple semi-supervised dependency parsing. *ACL-HLT*, 2008.

[21] J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking and the dynamics of the news cycle. In *KDD*, KDD '09, pages 497–506, 2009.

[22] X. Li and B. Liu. Learning to classify texts using positive and unlabeled data. In *IJCAI*, volume 3, pages 587–592, 2003.

[23] Y. Li, B.-J. P. Hsu, C. Zhai, and K. Wang. Unsupervised query segmentation using clickthrough for information retrieval. In *SIGIR*, 2011.

[24] Z. Liu, X. Chen, Y. Zheng, and M. Sun. Automatic keyphrase extraction by bridging vocabulary gap. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 135–144. Association for Computational Linguistics, 2011.

[25] R. McDonald, F. Pereira, K. Ribarov, and J. Hajič. Non-projective dependency parsing using spanning tree algorithms. In *EMNLP*, 2005.

[26] R. Mihalcea and P. Tarau. Textrank: Bringing order into texts. In *ACL*, 2004.

[27] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[28] D. P, A. Dey, and D. Majumdar. Fast mining of interesting phrases from subsets of text corpora. In *EDBT*, 2014.

[29] A. Parameswaran, H. Garcia-Molina, and A. Rajaraman. Towards the web of concepts: Extracting concepts from large datasets. *Proceedings of the Very Large Data Bases Conference (VLDB)*, 3((1-2)), September 2010.

[30] Y. Park, R. J. Byrd, and B. K. Boguraev. Automatic glossary extraction: beyond terminology identification. In *COLING*, 2002.

[31] V. Punyakanok and D. Roth. The use of classifiers in sequential inference. In *NIPS*, 2001.

[32] C. Ramisch, A. Villavicencio, and C. Boitet. Multiword expressions in the wild? the mwetoolkit comes in handy. In *COLING*, pages 57–60, 2010.

[33] B. Settles. *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2012.

[34] A. Simitsis, A. Baid, Y. Sismanis, and B. Reinwald. Multidimensional content exploration. *Proc. VLDB Endow.*, 1(1):660–671, Aug. 2008.

[35] R. Sproat, W. Gale, C. Shih, and N. Chang. A stochastic finite-state word-segmentation algorithm for chinese. *Computational linguistics*, 22(3):377–404, 1996.

[36] B. Tan and F. Peng. Unsupervised query segmentation using generative language models and wikipedia. In *WWW*, 2008.

[37] E. F. Tjong Kim Sang and S. Buchholz. Introduction to the conll-2000 shared task: Chunking. In *CONLL*, 2000.

[38] I. H. Witten, G. W. Paynter, E. Frank, C. Gutwin, and C. G. Nevill-Manning. Kea: Practical automatic keyphrase extraction. In *Proceedings of the fourth ACM conference on Digital libraries*, pages 254–255. ACM, 1999.

[39] E. Xun, C. Huang, and M. Zhou. A unified statistical model for the identification of english basenp. In *ACL*, 2000.

[40] D. Zhang, C. Zhai, and J. Han. Topic Cube: Topic Modeling for OLAP on Multidimensional Text Databases. In *SDM*, pages 1123–1134, 2009.

[41] Z. Zhang, J. Iria, C. A. Brewster, and F. Ciravegna. A comparative evaluation of term recognition algorithms. *LREC*, 2008.

8. APPENDIX

8.1 Adapted Aho-Corasick Automaton

The Aho-Corasick Automaton is similar to the data structure Trie, which could utilize the common prefix to save the memory usage and also make the process more efficient. It also computes the field "failed" referring to the node which could continue the matching process. In this paper, we adopt standard Aho-Corasick Automaton definition and construction process. Algorithm 5 introduced a "while" loop to fix the

Algorithm 5: Locating using Aho-Corasick Automaton

Input: The Aho-Corasick Automaton root, the corpus string $\mathcal{C}$.
Output: All occurrences of phrases in the automaton $\mathcal{O}$.

```
O ← ∅
u ← root
for i ← 1 to |C| do
    while u ≠ root and C[i] not in u.next do
        u ← u.failed
    if C[i] in u.next then
        u ← u.next[C[i]]
    p ← u
    while p ≠ root and p.isEnd do
        O ← O ∪ [i - p.depth + 1, i]
        p ← p.failed
return O
```

Table 8: Runtime of locating phrase candidates

	Academia	Yelp
Aho-Corasick Automaton	154.25s	198.39s
Unordered Map (Hash Table)	192.71s	366.67s

issues brought by prefix (*i.e.*, there might be some phrase candidates which are the prefix of the others), which is slightly different from the traditional matching process and could help us find all occurrences of the phrase candidates in the corpus in a linear time.

An alternative way is to adopt the hash table. However, one should carefully choose the hash function for hash tale and the theoretical time complexity of hash table is not exactly linear. For the comparison, we implemented a hash table approach using the unordered map in C++, while the Aho-Corasick Automaton was coded in C++ too. The results can be found in Table 8 We can see that Aho-Corasick Automaton is slightly better because of its exact linear complexity and less memory overhead.

8.2 Convergence Study of Viterbi Training

Our time complexity analysis in Sec. 4.5 assumes Viterbi Training in Alg. 3 converges in few iterations. Here we verify this through empirical studies. From Table 9, VT converges extremely fast on both datasets. This owes to the good initialization based on raw frequency as well as the particular property of Viterbi Training discussed in Sec. 4.3.

8.3 Iterations of SegPhrase+

SegPhrase+ involves only one iteration of re-estimating phrase quality using normalized rectified frequency from phrasal segmentation. Here we show the performance of SegPhrase+ with more iterations in Fig. 5 based on human-labeled phrases. For comparison, we also report performance of ClassPhrase+ which is similar with ClassPhrase but contains segmentation feature generated by results of phrasal segmentation from the last iteration.

We can see that the benefits brought by rectified frequency can be fully digested within the first iteration, leaving F1 scores over the next several iterations close. One can also

Table 9: Objective function values of Viterbi Training for SegPhrase and SegPhrase+

Dataset	Academia		Yelp	
Method	SegPhrase	SegPhrase+	SegPhrase	SegPhrase+
Iter.1	-6.39453E+08	-6.33064E+08	-9.33899E+08	-9.27055E+08
Iter.2	-6.23699E+08	-6.17419E+08	-9.12082E+08	-9.06041E+08
Iter.3	-6.23383E+08	-6.17214E+08	-9.11835E+08	-9.05946E+08
Iter.4	-6.23354E+08	-6.17196E+08	-9.11819E+08	-9.05940E+08
Iter.5	-6.23351E+08	-6.17195E+08	-9.11818E+08	-9.05940E+08

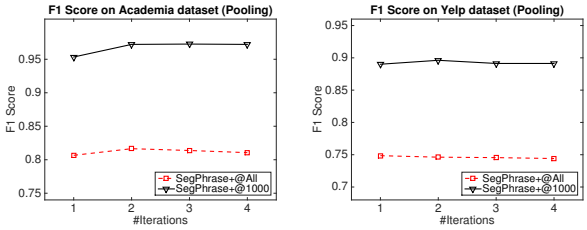

Figure 5: Performance variations of SegPhrase+ and ClassPhrase+ with increasing iterations

observe a slight performance decline over the next two iterations especially for the top-1000 phrases. Recall that we are reusing training labels for each iteration. Then this decline can be well explained by overfitting because segmentation features added by later iterations become less meaningful. Meanwhile, more meaningless features will undermine the classification power of random forest. Based on this, we can conclude that there is no need to do the phrase quality re-estimation multiple times.

8.4 Experiments on Larger Datasets

Since both Yelp and Academia dataset are relatively small in file size, we additionally tested our framework's running time on a larger data set composed of all Wikipedia articles as well as its performance considering all Wiki phrases to be ground truth. The statistics of Wikipedia dataset is shown in the following table and one can verify the linearality of computational complexity compared with previous experiments on Academia and Yelp reported in Table 10. The column *#wiki* shows the number of correctly predicted Wiki phrases of SegPhrase+. The precision-recall curves are reported in Fig. 6. Similar patterns can be observed compared to Fig. 1 that SegPhrase+ beats the other two proposed baselines and integration of phrasal segmentation with phrase mining boosts performance a lot.

Table 10: Statistics of experiments on Wiki dataset

file size	#words	time	#wiki phr.	# total
20.23GB	3.26G	28.08h	718,840	1,390,632

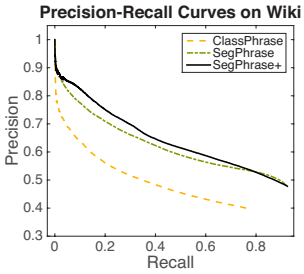

Figure 6: Precision-recall curves of the 3 proposed methods tested on Wikipedia dataset

8.5 Interesting Phrase Mining (More)

We show comparisons between SegPhrase+ and TopMine in Table 11. TopMine approaches phrase mining based on raw frequency as well as document context following a bottom-up fashion. Such formulation can be interpreted as a greedy segmentation by iteratively merging most correlated units in a sentence. Such a greedy segmentation is not optimal compared to our segmentation framework using dynamic programming. Moreover, TopMine removes stopwords before greedy segmentation. This loses important information

Table 11: Interesting phrases mined from SIGMOD and SIGKDD conferences (SegPhrase+ v.s. TopMine)

Conference	SIGMOD		SIGKDD	
Method	SegPhrase+	TopMine	SegPhrase+	TopMine
1	data base	data base	data mining	data mining
2	database system	query optimization	data set	data set
3	relational database	database system	association rule	knowledge discovery
...	...	...	...	...
51	sql server	aggregation function	association rule mining	data item
52	relational data	query answering	rule set	user profile
53	data structure	query rewriting	concept drift	user preference
54	join query	data access	knowledge acquisition	rule discovery
55	web service	hash join	gene expression data	sequential pattern
...	...	...	...	...
201	high dimensional data	logical and physical	active learning	
202	location based service	transitive closure	frequent subgraph	algorithm is efficient
203	xml schema	performance study	intrusion detection	approach is proposed
204	two phase locking	disk access	categorical attribute	data cube
205	deep web	object oriented	user preference	distance metric
...	...	...	...	...

Table 12: Interesting phrases mined from SIGMOD and SIGKDD conferences (SegPhrase+ v.s. SegPhrase)

Conference	SIGMOD		SIGKDD	
Method	SegPhrase+	SegPhrase	SegPhrase+	SegPhrase
1	data base	data base	data mining	data mining
2	database system	database system	data set	data set
3	relational database	query processing	association rule	association rule
...	...	...	...	...
51	sql server	subsequence matching	association rule mining	association rule mining
52	relational data	sql server	rule set	rule set
53	data structure	relational data	concept drift	concept drift
54	join query	data structure	knowledge acquisition	knowledge acquisition
55	web service	web service	gene expression data	gene expression data
...	...	...	...	...
201	high dimensional data	query containment	web content	web search engine
202	location based service	buffer management	frequent subgraph	better than
203	xml schema	garbage collection	intrusion detection	previously proposed
204	two phase locking	moving object	categorical attribute	application domain
205	deep web	on demand	user preference	wide range
...	...	...	...	...

because stopwords can be naturally serving as boundaries between meaningful phrases. From the table we can exactly tell that interesting phrases mined from our segmented corpus are better.

Comparisons between SegPhrase+ and SegPhrase are in Table 12. The results are basically similar but we can notice that phrases generated based on SegPhrase+'s segmentation are better in quality than SegPhrase's.

8.6 Relevant Word/Phrase Mining (More)

We show comparisons between SegPhrase+ and TopMine in Table 13, SegPhrase+ and SegPhrase in Table 14. From these two tables one can tell that SegPhrase+ beats TopMine while behaves similar to SegPhrase. This is reasonable because we are only listing the most similar words/phrases. As SegPhrase+ and SegPhrase are sharing the same phrasal segmentation module, the differences among top-ranked phrases will not be significant.

8.7 Case Study of Quality Phrases

We show some phrases from ranking lists generated by ClassPhrase, SegPhrase and SegPhrase+ in Table 15. In general, phrase quality drops with number goes up. ClassPhrase always performs the worst among the three methods. SegPhrase+ is slightly better than SegPhrase, which is noticeable for phrases ranked after 20,000. It's worth mentioning that the minimum sizes of phrase lists are 50,577 and 42,989 for two datasets respectively.

8.8 Error Analysis

We analyze of errors of SegPhrase+ from the aspects of false positive and false negative.

- False positive (reflected in precision): Through our studies, phrases with poor quality are mistakenly predicted as good ones due to the fixed collocation among words. For example, 'experiments are conducted' gets 0.88 quality score and 'at the network layer' gets 0.77. Such cases become more frequent when the phrases are close to the bottom of the ranking list.
- False negative (reflected in recall): Quality phrases are underestimated because their frequencies in the corpus is significantly lower than those of their sub-components. For example, 'topic model's score is 0.44, compared to 'topic modeling' which gets 0.72. As 'topic' and 'model' are both very frequent in the corpus, it is more difficult for 'topic model' to be predicted as quality phrase.

Based on the above error analysis, there are three inspired directions to improve the model. The first is to modify the quality estimator by incorporating syntactic features obtained from a pre-trained POS tagger. The second is to improve the viterbi training module in phrasal segmentation. One alternative is to find top-k segmentations instead of the best one given a text snippet. This is less deterministic in performing segmentation than the current design but consumes more computational power. Meanwhile, segment length penalty can be designed in a more sophisticated way by dynamically changing its value based on the local context instead of setting a global value. However, this will inevitably require more labels or predefined rules. The third aspect is to do normalization on words in the corpus such as stemming. This may merge 'topic modeling' and 'topic model' and make their predictions consistent.

Table 13: Top-5 similar words/phrases for queries from Academia/Yelp datasets (SegPhrase+ v.s. TopMine)

Query	data mining		olap	
Method	SegPhrase+	TopMine	SegPhrase+	TopMine
1	knowledge discovery	knowledge discovery	data warehouse	olap system
2	text mining	integrating data mining	online analytical processing	online analytical processing olap
3	web mining	data mining tools	data cube	olap queries
4	machine learning	mining	olap queries	data cube
5	data mining techniques	data mining dm	multidimensional databases	queries in olap

Query	blu-ray		noodle		valet parking	
Method	SegPhrase+	TopMine	SegPhrase+	TopMine	SegPhrase+	TopMine
1	dvd	movie	ramen	noodles the noodles	valet	valet
2	vhs	recordable	noodle soup	noodles weren	self-parking	valet service
3	cd	adobe's	rice noodle	noodle dish	valet service	parking and valet
4	new release	itunes	egg noodle	rice noodles	free valet parking	free valet
5	sony	bootable	pasta	noodle texture	covered parking	valet your car

Table 14: Top-5 similar words/phrases for queries from Academia/Yelp datasets (SegPhrase+ v.s. SegPhrase)

Query	data mining		olap	
Method	SegPhrase+	SegPhrase	SegPhrase+	SegPhrase
1	knowledge discovery	knowledge discovery	data warehouse	data warehouse
2	text mining	text mining	online analytical processing	online analytical processing
3	web mining	machine learning	data cube	data cube
4	machine learning	data mining techniques	olap queries	multidimensional database
5	data mining techniques	mining	multidimensional databases	olap operations

Query	blu-ray		noodle		valet parking	
Method	SegPhrase+	TopMine	SegPhrase+	TopMine	SegPhrase+	TopMine
1	dvd	dvd	ramen	noodle soup	valet	valet
2	vhs	new release	noodle soup	egg noodle	self-parking	valet service
3	cd	on netflix	rice noodle	ramen	valet service	self parking
4	new release	vhs	egg noodle	rice noodles	free valet parking	complimentary valet
5	sony	cd	pasta	noodle and	covered parking	parking

Table 15: Sampled quality phrases from Academia and Yelp datasets

Method	ClassPhrase	SegPhrase	SegPhrase+
1	virtual reality	virtual reality	self organization
2	variable bit rate	variable bit rate	polynomial time approx. scheme
3	shortest path	shortest path	least squares
...	...	...	...
501	finite state	frequency offset estimation	health care
502	air traffic	collaborative filtering	gene expression
503	long term	ultra wide band	finite state transducers
...	...	...	...
10001	search terms	test plan	airline crew scheduling
10002	high dimensional space	automatic text	integer programming
10003	delay variation	adaptive bandwidth	web log data
...	...	...	...
20001	test coverage	implementation costs	experience sampling
20002	adaptive sliding mode control	error bounded	virtual execution environments
20003	random graph models	free market	nonlinear time delay systems
...	...	...	...
50001	svm method	harmony search algorithm	asymptotic theory
50002	interface adaptation	integer variables	physical mapping
50003	diagnostic fault simulation	nonlinear oscillators	distinct patterns
...	...	...	...

Method	ClassPhrase	SegPhrase	SegPhrase+
1	taco bell	taco bell	tour guide
2	wet republic	wet republic	yellow tail
3	pizzeria bianco	pizzeria bianco	vanilla bean
...	...	...	...
501	panoramic view	art museum	rm seafood
502	pretzel bun	ice cream parlor	pecan pie
503	spa pedicure	pho kim long	master bedroom
...	...	...	...
10001	seated promptly	carrot soup	gary danko
10002	leisurely stroll	veggie soup	benny benassi
10003	flavored water	pork burrito	big eaters
...	...	...	...
20001	buttery toast	late night specials	cilantro hummus
20002	quick breakfast	older women	lv convention center
20003	slightly higher	worth noting	iced vanilla
...	...	...	...
40001	friday morning	conveniently placed	coupled with
40002	start feeling	cant remember	way too high
40003	immediately start	stereo system	almost guaranteed
...	...	...	...