Article

JIS

Journal of Information Science
38(5) 476–488

DOI: 10.1177/0165551512448984
jis.sagepub.com
SAGE

Keyphrase extraction through query performance prediction

Gonenc Ercan
Bilkent University, Turkey

Ilyas Cicekli
Hacettepe University, Turkey

Abstract
Previous research shows that keyphrases are useful tools in document retrieval and navigation. While these point to a relation between keyphrases and document retrieval performance, no other work uses this relationship to identify keyphrases of a given document. This work aims to establish a link between the problems of query performance prediction (QPP) and keyphrase extraction. To this end, features used in QPP are evaluated in keyphrase extraction using a naïve Bayes classifier. Our experiments indicate that these features improve the effectiveness of keyphrase extraction in documents of different length. More importantly, commonly used features of frequency and first position in text perform poorly on shorter documents, whereas QPP features are more robust and achieve better results.

Keywords
keyphrase extraction; query performance prediction

1. Introduction

Keyphrases are phrases formed of one or more terms that reflect the salient content of a document. The term 'keyphrase' is used instead of the more common expression 'keyword' to emphasize that keyphrases can be formed of multiple terms. A user attempting to fulfil an information need issues a query to a full-text search engine and scans each retrieved document to have at least a basic understanding of its contents. In this context, keyphrases are helpful in terms of shortening the time spent, as they are concise representations of the content. Nonetheless, most of the electronically available documents do not have keyphrases assigned to them. One approach to this problem is to build a keyphrase extraction system that automatically determines the keyphrases for a given document.

A typical keyphrase extraction system forms a list of candidates from the phrases that appear in the original document, and evaluates each candidate using features acquired from the text. Two commonly used features are the frequency and first occurrence position of the phrase, which are referred to as in-document features in this article. This research shows, through experiments, that in-document features are not as effective in short documents as they are in long documents. Following from this, it proposes a method using novel features to achieve more robust results for documents of different lengths.

Our method is motivated by the observation that, when a keyphrase is searched in a corpus, the retrieved set of documents is relatively focused on a single domain with high similarity. For example, while 'machine' and 'learning' are two ambiguous terms that appear in documents from different domains, the phrase 'machine learning' appears in a document set that is more concentrated on a single domain. In this context, keyphrases are expected to be good queries, and are utilized in document retrieval and browsing. Phrasier [1] creates an index using only the keyphrases of documents instead of the full text, and reports no negative impact on document retrieval performance. Furthermore, Gutwin et al. [2] discuss

Corresponding author:
Gonenc Ercan, Department of Computer Engineering, Bilkent University P.O. 06800 Ankara Turkey.
Email: ercangu@cs.bilkent.edu.tr

the use of keyphrases as a tool for browsing a digital library. The results of both works indicate that keyphrases perform well in document retrieval. This has encouraged us to determine a phrase's keyphraseness by measuring its query performance in document retrieval.

Ideally, evaluating the retrieval performance of a query requires manual judgements of each returned document's relevance to the user's information need. As it is not possible to obtain relevance judgements for all possible queries, a prediction of query performance is used instead. Query performance prediction (QPP) [3] is a research problem in information retrieval that aims to estimate how good a query is. That is to say, in QPP a well-performing query is expected to retrieve a document set relevant to the topic that the user is interested in. If a query issued to retrieve documents about topic A retrieves documents for two different topics A and B, then documents relevant to topic B are irrelevant, and this degrades the query performance. As in this example, ambiguous queries retrieving documents from different topics are not good queries, since they present documents irrelevant to the user's information need. This is very similar to our observation about keyphrases, that a keyphrase should represent a document's content clearly, without any ambiguity.

QPP gained popularity as its applications proved beneficial for search engines by improving document retrieval performance. Its major applications are selective query expansion [3, 4], merging results in a distributed retrieval system, and missing content detection [5] to improve the efficiency and effectiveness of document retrieval systems. This work contributes to the keyphrase extraction literature by showing that methods used in QPP improve keyphrase extraction. Most of the previous research uses features that are intrinsic properties of the given document, whereas the features introduced in this article are calculated using a background corpus. Furthermore, we show empirically that QPP features are more effective in shorter documents, compared to in-document features.

Section 2 introduces the related works on keyphrase extraction. Following this, in Section 3 we give the details of our algorithm. In Section 4 we present the corpus used in our experiments, the evaluation metrics, and the results. Finally, we conclude in Section 5 with a discussion of the results and possible future work.

2. Related works

Keyphrase extraction algorithms typically score each candidate phrase with a keyphraseness scoring function, and sort the candidates accordingly. This function is implemented by either a supervised or an unsupervised learning algorithm. Supervised keyphrase extraction algorithms train a keyphraseness function automatically from observations in a corpus, whereas unsupervised keyphrase extraction algorithms depend on scoring functions built on assumptions and observations. Our keyphrase extraction algorithm is a supervised learning algorithm that uses QPP features.

GenEx [6] is a supervised keyphrase extraction algorithm formed of two components: an extractor and a genetic algorithm. The extractor is a text-processing tool controlled by parameters and rules. For example, the aggressiveness of the stemmer and maximum number of the terms allowed in an extracted keyphrase are two parameters of the extractor. GenEx uses a genetic algorithm and a training set to find the most suitable parameter values for a domain. The population of the genetic algorithm is formed of parameter sets representing a configuration of the extractor. The fitness function is the precision of the extractor executed with the processed parameter-set/configuration. The output of the genetic algorithm is the set of rules suitable for the corpus domain and genre. GenEx uses the frequency and first occurrence position of terms in order to score each phrase's importance.

The Kea algorithm uses naïve Bayes to learn a keyphraseness probability from the in-document features' distribution in the training data. The outline of the Kea algorithm is identical to the outline of the system used in our experiments. Frank et al. [7] report the results of Kea and GenEx [6] to be statistically indifferent. We also use naïve Bayes in order to learn the keyphraseness probabilities of our QPP features. In order to evaluate the QPP features in a systematic manner, we compare our method with the Kea [7] algorithm's effectiveness.

Recent research aims to improve the effectiveness of keyphrase extraction by integrating additional features such as those exploiting the structural patterns of a document, syntactic patterns of phrases, semantic knowledge, and the relationships between extracted keyphrases. For instance, syntactic features are able to eliminate candidate phrases that are unlikely to be keyphrases (as in the example of the phrase 'machine learning introduces', which ends with a verb, and is uncommon for a keyphrase). Hulth [8] investigates the effectiveness of using part of speech (PoS) tags in keyphrase extraction. She evaluates several candidate phrase extraction strategies with and without PoS tags. Using a rule induction method, PoS tag patterns for keyphrases are learned. Hulth's experiments indicate that keyphrases have common PoS tag patterns. Furthermore, Barker et al. [9] use a chunker to detect noun phrases using simple syntactic patterns, and assign a score to each phrase depending on the *tf***idf* value of each phrase's head noun. The in-document features are used to evaluate phrases extracted from the document by a noun phrase chunker. Recent work [10–12], including ours, uses syntactic filters in finding the candidate keyphrases.

The cohesive ties between keyphrases are exploited in keyphrase extraction by using external knowledge obtained either from thesauri or from statistical information extracted from corpora. Turney [13] notes that there should be cohesion between the extracted keyphrases. He measures the pairwise semantic relatedness between two keyphrases by mining the results of a search engine. The cohesion between the keyphrases produced by Kea is reclassified with a second classifier using the cohesion features. This method depends on the output of Kea and in-document features. For extracting keywords, Ercan and Cicekli [14] use the WordNet thesaurus [15] to integrate semantic relationships between phrases and features extracted from the lexical chains of the original document. Thus their method is not able to handle keyphrases of length greater than one, and is limited to keyword extraction. Nguyen et al. [10] integrate both PoS tags and structural features in the feature set of Kea, where the distinguished structural properties of research articles are important cues for keyphrase extraction. They use the occurrence distribution of phrases with respect to the sections of the processed article. For example, a phrase appearing in the title or abstract is more likely to be a keyphrase than a phrase appearing only in the middle of the document. This method is designed specifically for research articles, and is not a generic solution.

Mihalcea et al. [11] model the text as a graph, where the vertices are terms appearing in the given text, and an edge exists between two terms if they co-occur within a distance. Keyphrases are extracted using a social ranking algorithm called TextRank on this network, which is based on the PageRank algorithm [16]. The phrase co-occurrence graph is usually very sparse, especially in short documents where term frequencies are low. Recently, Wan et al. [12] have enriched this graph by integrating co-occurrence statistics observed in similar documents, that is, the nearest neighbours of the processed text (NN-TextRank). They evaluate their algorithm using news articles that are shorter than research articles, and report improvement over TextRank and a baseline algorithm that scores phrases using *tf*idf* values. Although both QPP features and NN-TextRank use a background corpus in order to handle short documents, these algorithms differ both in their methods and in the features applied.

3. Keyphrase extraction

The general components of our supervised keyphrase extraction system are depicted in Figure 1. These are similar to some of those in the previous works [6–8, 10], as they all perform feature extraction and supervised classification. The first step in keyphrase extraction is tokenization of the text into words and punctuations. Using the token stream, a candidate keyphrase list is formed from the phrases appearing in the text of the original document. For each candidate phrase w the feature extraction component extracts a feature set from the background corpus and the original document d_0. A naïve Bayes classifier trained with documents of the same genre and their associated keyphrase lists calculates the probability of keyphraseness for each candidate phrase w. Keyphrases are selected using these scores, and the output of the system is a set of keyphrases. The keyphrase selection component simply sorts the phrases according to the keyphraseness score and returns the top keyphrases.

The feature extraction component is what distinguishes our keyphrase extraction system from the others. The feature extractor uses a background corpus to determine some intrinsic properties of each phrase w, as depicted in the flowchart in Figure 2. The feature extractor operates on a phrase w, finding the document set D' formed of all the documents that w appears in. Both for efficiency and for effectiveness, a subset of D' is selected by a sampling procedure. Features are extracted from this subset D and the original document d_0.

3.1. Candidate keyphrase list creation

Keyphrases usually appear as noun phrases in documents. Hulth [8] reports that nouns preceded by nouns or adjectives are the most common part of speech (PoS) tag patterns observed for keyphrases. In fact, the majority of keyphrases in our corpora can be extracted with a simple grammar rule for finding noun phrases. In our system, in order to create a candidate keyphrase list, the text of the given document d_0 is tokenized, and the PoS tags are assigned using the Stanford PoS tagger [17]. Each sequence of PoS tags satisfying the regular expression *'(JJ|NN) * NN'* is included in the candidate phrase list, where *NN* represents nouns and *JJ* represents adjectives. For example, in the sentence 'This is a good\JJ machine\NN learning\NN algorithm\NN'; 'good machine learning algorithm', 'good machine learning', 'machine learning algorithm', 'machine learning', 'learning algorithm', 'machine', 'learning', 'algorithm' match the regular expression and are extracted as candidate phrases. Only the candidate keyphrases matching the regular expression are retained and evaluated by the classification algorithm.

The PoS pattern method can detect more than 80% of all keyphrases in the research article corpus used in our experiments as candidate keyphrases. We compared this method with an exhaustive candidate keyphrase extraction method, which returns all consecutive terms that do not contain punctuations as candidate keyphrases. The exhaustive method

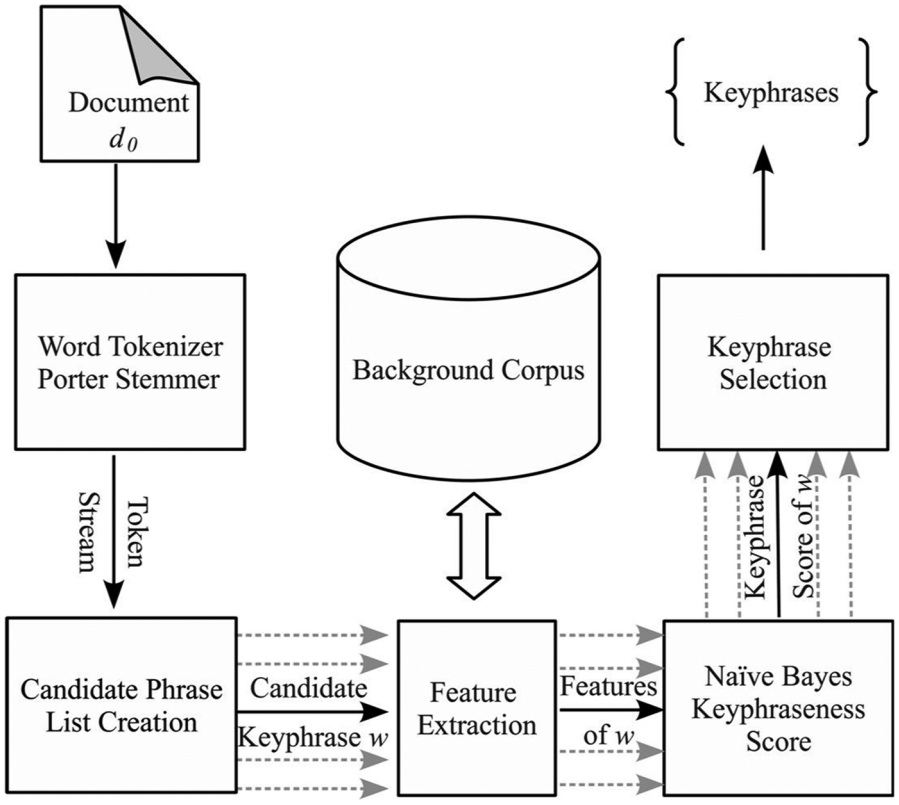

Figure 1. Components of the keyphrase extraction system.

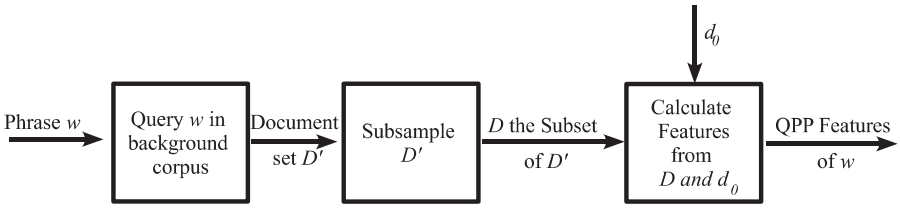

Figure 2. Flowchart of feature extractor.

extracts all the keyphrases appearing in the text as candidate keyphrases. It detects 82% of the keyphrases in the research article corpus, which is only 2% more than from the PoS pattern method. However, this method produces more candidates than the PoS pattern method where most candidates are unlikely to be keyphrases or even phrases. For example, in the corpus the exhaustive method produces approximately 35,000 candidate phrases whereas the PoS method produces approximately 2000 candidates per document. When the former method is used, the system must process a larger number of candidate keyphrases, which degrades the efficiency of the system. In addition, the effectiveness of the system is not improved, as having more candidate keyphrases creates noise for the classification algorithm. Since the PoS pattern method is as effective as the exhaustive method, we opted to use the PoS pattern method as our candidate phrase producer in our keyphrase extraction system.

3.2. Information retrieval from Wikipedia

The information retrieval component (IRC) performs full-text search in the background corpus, which is composed of Wikipedia articles. Since the keyphrase extraction algorithm processes the document vectors and language models of retrieved documents by accessing their full texts, an offline indexing system is preferred. Document retrieval is an

important factor for both the efficiency and the effectiveness of the keyphrase extraction system. This section describes how full-text search is performed from the background corpus for a candidate keyphrase given as a query, and how the returned documents are subsampled.

Given a term as a query, the IRC returns the set of articles that contain the searched term. Given a phrase (i.e. multiple terms) as a query, the IRC returns the set of articles in which the terms of the phrase appear in exactly the same configuration. In order for an IRC to support such phrase queries, either a phrase or a positional index must be maintained [18]. For example, for the phrase 'machine learning', documents containing 'machine learning' are returned, whereas those containing 'learning machine' are not. Implementation of the IRC uses the indexing mechanisms of the Lucene search engine.[1] A Lucene inverted index supports positional indices. For each term, a sorted list of articles containing the term and its position within the text are stored.

Given a phrase w formed of one or more terms, let $D' = \{d_1, d_2, \ldots, d_m\}$ be the set of documents that contain w. If w occurs at least once in all indexed articles of the background corpus, the size of D' may be as large as the length of the collection queried. Since in-depth analysis of the m returned documents is not efficient, and may result in noise due to variations in $|D'|$ for different phrases, a subset D is sampled from D'. The sample size μ is a parameter of the system, and the size of the sample set denoted as $|D|$ is $\min(|D'|, \mu)$. Higher values of μ increase the variance of $|D|$ for different phrases, and thus create a bias towards phrases that appear in a smaller number of documents in the background corpus. A lower value of μ is not able to represent the set $|D'|$. Our empirical evaluations show that using a sample size $\mu = 20$ avoids problems caused both by the variations in $|D|$ and by the underrepresentation of the set D'.

Two sampling strategies were evaluated: random and rank-based. Random sampling simply selects μ documents from D' randomly. Rank-based sampling reduces the noise caused by documents with low frequencies of w. Accordingly, documents are ranked using a function of the frequency of phrase w, and only the μ top-scoring documents are retained. Since the ranked sampling strategy achieves better results than the uniform sampling strategy, only the result of the ranked sampling strategy is reported in this article. In our experiments, the Okapi BM25 ranking function is used:

$$\mathrm{BM25}(w,d_i) = \frac{c(w,d_i)(k_1+1)}{c(w,d_i) + k_1[1 - b + b(|d_i|/avgDocLen)]} \tag{1}$$

where $c(w, d_i)$ is the frequency of the phrase w in the document d_i, and the parameters b and k_1 control the scoring function's behaviour: b controls the weight of the document length, and k_1 controls the weight of term frequency in the ranking. The values $k_1 = 2$ and $b = 0.75$ are used in our experiments, as Jones [1] reports that these values correlate with human relevance judgements. The average document length in the background corpus is denoted by *avgDocLen*, and $|d_i|$ denotes the length of the document d_i.[2]

The background corpus used in our experiments is composed of English Wikipedia[3] articles that are longer than 200 characters. The number of Wikipedia articles indexed is 3,326,028, containing a total of approximately 1 billion terms. The average document length of the corpus is 237.89 non-stop words.[4] Wikipedia is a generic background corpus, since it is a comprehensive encyclopaedia relating to different topics. In practice, the background corpus can be domain specific. For instance, in a digital library, an index of all the articles stored can be preferred instead of the generic Wikipedia articles corpus. We chose to use Wikipedia to have a domain-independent system.

3.3. QPP measures

This section defines the QPP measures utilized in this article on the basis of the assumption that keyphrases are unambiguous query phrases that retrieve documents tied to each other by a specific domain or topic. Note that this assumption is also important for a retrieval system. For example, for the query 'learning' a diverse set of documents is returned, whereas for the query 'machine learning' a more refined set of documents is returned. The QPP problem [3] tries to predict the effectiveness of a query, and should encourage the query 'machine learning' over 'learning'. The experiments discussed in this paper evaluate the effects of different QPP techniques [3, 4, 19–22] in keyphrase extraction.

Table 1 lists the features used in keyphrase extraction. The first two of these features, *firstPosition* and *tf*idf*, are in-document features, and the last seven are QPP features, which can be grouped in two categories depending on the methods used in their extraction. The first category uses the geometrical properties of the retrieved documents when represented by vector space models (VSM). The second class of methods uses ideas from language models (LM) and information theory. The extraction of each feature is explained for a single phrase w, using two inputs: the document set D and the original document d_0. Features from 3 to 7 are calculated using vector space models, and the last two are based on language models.

Table 1. Features used in keyphrase extraction

ID	Feature name	Value range	Description
1	*firstPosition*	[0, 1]	Distance of first occurrence of phrase from top of d_0
2	*tf*idf*	[0, ∞]	Term frequency times inverse document frequency
3	$CosCentrTod_0$	[0, 1]	Cosine similarity of centroid of D to d_0
4	$avgCosTod_0$	[0, 1]	Average cosine similarity of documents in D to d_0
5	*interSim*	[0, 1]	Mean of pairwise cosine similarities of documents in D
6	*CoxLewisTest*	[0, 1]	Cox–Lewis clustering tendency test
7	*DocPertub*	$[-\|D\|, \|D\|]$	Average rank change in D under document perturbation
8	*Clarity*	[0, ∞]	KL divergence of D language model from background corpora language model
9	$KLDocsFromd_0$	[0, ∞]	KL divergence of D language model from d_0 language model

Both VSM-based and LM based methods use a bag-of-words assumption to simplify the analysis. In order to create a bag or set of words, documents are tokenized to words (terms). Words are processed with a Porter stemmer [23] to conflate inflected forms of words with their base forms. In our presentations, the set V denotes the vocabulary formed of conflated words occurring in the background corpus and d_0, the function $c(t_k, d_i)$ returns the number of occurrences of the kth word of V in the ith document, and f_k is the number of documents containing the word t_k. The retrieved documents in the set D and the original document d_0 are tokenized using this method.

3.3.1. Vector space model based features. A vector space model defines each document as a vector in a $|V|$-dimensional space, where each dimension corresponds to a word in the vocabulary V. Let d_{ik} represent the kth term's weight in the ith document, where the original document is indexed by 0 and the documents in the set D are indexed from 1 to $|D|$. The weight d_{ik} is calculated as shown in equation (2), where N is the number of documents in the background corpora, and f_k is the number of documents in which the kth term occurs. The weighting function d_{ik} is the term frequency ($tf = c(t_k,d_i)$) times inverse document frequency ($idf = \log(N/f_k)$), and is termed the *tf*idf* weighting.

$$d_{ik} = c(t_k,d_i)\log\frac{N}{f_k} \tag{2}$$

A document vector d_i consists of the weights of all terms of the vocabulary V in the ith document. Two document vectors can be compared with each other through different similarity metrics. The cosine of the angle between two vectors is one such similarity function, called the cosine similarity. The cosine similarity between documents d_i and d_j is calculated using the equation

$$cosSim(d_i,d_j) = \frac{\sum_{k=0}^{|V|} d_{ik}d_{jk}}{\sqrt{\sum_{k=0}^{|V|} d_{ik}^2}\sqrt{\sum_{k=0}^{|V|} d_{jk}^2}} \tag{3}$$

In this model, our assumption is that a keyphrase w has to retrieve a document set that is geometrically closer to d_0, cohesive, less scattered and more concentrated. When defined in terms of similarity, all documents in D and the original document d_0 should be similar to each other. The average similarity of the retrieved documents to d_0 ($avgCosTod_0$) is calculated using

$$avgCosTod_0 = \frac{1}{|D|}\sum_{i=1}^{|D|} cosSim(d_i,d_0) \tag{4}$$

The inter-similarity feature (*interSim*) calculates the average pairwise similarity of the retrieved documents using

$$interSim = \frac{2}{|D|(|D|-1)}\sum_{i=1}^{|D|}\sum_{j=i+1}^{|D|} cosSim(d_i,d_j) \tag{5}$$

Kwok et al. [25] use a similar metric to predict the performance of queries. Since cosine similarity function is symmetric, the average can be calculated using only $|D|(|D|-1)/2$ similarities.

Calculation of $avgCosTod_0$ requires pairwise similarity calculations of each document with document d_0. Another method used in text categorization and summarization [24, 25] calculates the centroid of documents, and uses the similarity to the centroid instead. The centroid of D, denoted by $\breve{D}$, is the arithmetic mean of the document vectors, and is calculated using

$$\breve{D} = \frac{1}{|D|}\sum_{i=1}^{|D|} d_i \tag{6}$$

The $CosCentrTod_0$ measure is just the cosine similarity of document d_0 and $\breve{D}$. $CosCentrTod_0$ and $avgCosTod_0$ are two similar measures. They are equal if all the input document vectors are unit vectors, that is, if the norms of vectors are 1. In our case they are not unit vectors, and thus these two values are not equal but only similar. The $CosCentrTod_0$ feature can be interpreted as a comparison of d_0 with a virtual document formed by concatenating all the retrieved documents in D. The $avgCosTod_0$ feature takes into account the local relationships between each retrieved document and d_0, whereas $CosCentrTod_0$ is based on a more global view of the term usage in D. Furthermore, our experiments have shown that using both $CosCentrTod_0$ and $avgCosTod_0$ together achieves the best results.

Although measuring the similarity between documents is a good indicator for QPP, a coherent set may not always have a high average similarity, because of outliers and noise in D. Vinay et al. [21] define three measures: the Cox–Lewis clustering tendency, query perturbation, and document perturbation. Through a modified version of the Cox–Lewis clustering tendency test, the first measure evaluates D for the existence of either natural groupings or randomness. Vinay et al. [21] introduce query and document perturbation. The former modifies the query issued by a random noise, and observes the rank change in retrieval results. In our work we apply this measure by using d_0 as the issued query. Vinay et al. [21] report that this measure is not able to predict the query performance effectively. In an affirming manner, we observe that this feature is not effective in keyphrase extraction. For this reason, we are not reporting the results of the query perturbation feature.

Different tests of clustering tendency exist in the literature. The Hopkins test [27] and the Cox–Lewis statistic [28] are two such tests in which the points in the original set and the randomly generated points are compared to determine the randomness of the set. If a higher similarity to random points is observed, then the original set is randomly distributed in the space.

These tests are suitable when there are only a few dimensions, but they are not directly applicable to high-dimensional hyperspaces. In a high-dimensional space such as $|V|$ dimensions, where V is typically in the order of thousands, a randomly generated point will most probably be distant from D as the probability space is large. In order to limit the probability space, Vinay et al. [21] propose using a document in D as a skeleton for the random generation, and avoid creating a random document composed of unlikely term combinations. The Cox–Lewis test selects a document randomly from D, and assigns random term weights to its non-zero dimensions to form a random document vector rd_i. Random weights are between 0 and the maximum term weight appearing in set D. This generation strategy keeps the randomly generated points in a minimal hyper-rectangle containing all the documents in D.

Let nd_{i1} denote the nearest neighbour of the random vector rd_i in D, and let nd_{i2} be the nearest neighbour of nd_{i1} in D. The proportion of the similarity $cosSim(nd_{i1}, nd_{i2})$ to $cosSim(nd_{i1}, rd_i)$ tests whether the injected random vector can be more similar to a document in D than any other document in D. This test is repeated with $|D|/2$ random numbers, and the average of these tests is used as the Cox–Lewis score:

$$CoxLewisTest = \sum_{i=0}^{|D|/2} \frac{cosSim(nd_{i1},nd_{i2})}{cosSim(nd_{i1},rd_i)} \tag{7}$$

Document perturbation was first described by Vinay et al. [21] using VSM, and has recently been adapted to language models as rank robustness [22]. Similar to the Cox–Lewis test, the effect of adding random noise to the documents in D is tested. Given the document set D, when the documents are in descending order according to $cosSim(d_i, d_j)$ values – that is, numbered from the most similar to the least – the function $rank(d_i, D, d_j)$ is the rank of document d_j with respect to document d_i. The value of $rank(d_i, D, d_i)$ is equal to 1 with similarity 1.0 when documents are unique in D. The test modifies d_i by adding noise, and checks whether the set D contains documents more similar than d_i to perturbed d_i ($noise(d_i, \alpha)$). In a document set formed of unrelated documents, the rank of d_i does not change, whereas in a coherent set, rank change is expected to be high. Let α be a parameter controlling $noise(d_i, \alpha)$, and the function $noise(d_i, \alpha)$ return a vector perturbed by adding noise to each dimension of vector d_i. The noise is generated using a Gaussian distribution with mean

equal to 0, and deviation equal to α. The overall rank change for a noise level is calculated by repeating the test 10 times for all documents in D

$$docPerturb(\alpha) = \sum_{i=1}^{|D|} \sum_{0}^{10} rank[noise(d_i,\alpha),D,d_i] \tag{8}$$

The overall *docPerturb* feature is the slope of the line that best fits the $docPerturb(\alpha)$ values. The document perturbation test uses the noise levels $\alpha = \{0.1, 1, 10, 100\}$. If the slope is positive, then the rank changes as the noise level increases, and the set is assumed to be coherent.

3.3.2. Language model based features. Language models are used in different applications of information retrieval research [28, 29]. Unigram language models are formulated by a bag-of-words assumption, and ordering of words is not taken into account. A simple estimate of the probability of generating a term t_i from a document d_j is the maximum likelihood estimate (MLE) – that is, the relative frequency of t_i in d_j.

MLE usually results in a probability distribution with sharp changes, which assigns zero probability to terms not appearing in the document. Smoothing is a technique applied to resolve these problems. The probabilities of low or non-occurring terms are increased, and the probabilities of frequent terms are degraded. Jelinek–Mercer smoothing combines the MLE of the whole document collection with a document's MLE, providing a smoother probability distribution. Two language models are combined by a weighted average controlled by the parameter λ. We used the same value as utilized in Townsend et al. [3], which is 0.6. The linear combination of MLE of a document d_j with the whole background collection (all the Wikipedia articles) is given by

$$P(t|d_j) = \lambda P_{ML}(t|d_j) + (\lambda - 1)P_{ML}(t|Wiki) \tag{9}$$

With the above probability estimate for each term, we derive a simplified clarity score motivated by the score defined by Townsend et al. [3]. The relevance of a term t to the query phrase w and original document d_0, $P(t|w, d_j)$, is modelled by

$$P(t|w,D) = \sum_{j=1}^{|D|} P(t|d_j)P(d_j) \tag{10}$$

where $P(t \mid d_j)$ reflects the probability of observing term t in the document d_j in the set D. The probability $P(d_j)$ is uniform for all documents in D, and is equal to $1/|D|$.

The clarity measure is the Kullback–Leibler (KL) divergence [31] of the retrieved set D from the whole background corpus, defined as

$$Clarity = \sum_{t \in D} P(t|w,D)\log\left[\frac{P(t|w,D)}{P_{ML}(t|Wiki)}\right] \tag{11}$$

KL divergence compares two different probability models. It is used as a document similarity function in various information retrieval tasks, such as text clustering and categorization [32, 33]. In a similar fashion to $CosCentrTod_0$ and $avgCosTod_0$ features, the relationship of the retrieved set D to the original document d_0 is investigated using the $KLDocsFromd_0$ feature. This feature is simply the KL divergence of the language model $P(t|w,D)$ from d_0, calculated according to

$$KLDocsFromd_0 = \sum_{t \in D} P(t|w,D)\log\left[\frac{P(t|w,D)}{P_{ML}(t|d_0)}\right] \tag{12}$$

3.4. Learning to classify keyphrases

Keyphrase extraction can be considered as a classification task with two classes: keyphrase or non-keyphrase. For a specific domain or genre, a supervised machine learning algorithm analyses, learns and classifies keyphrases. Previous work on keyphrase extraction suggests that different types of corpora behave differently, and thus should be trained for each applied domain [6, 7].

The naïve Bayes learning algorithm uses a Bayesian rule to infer the probability of class membership, given the features. Using the independence assumption, the probability of keyphraseness $P(keyphrase \mid F_w)$ is calculated, where F_w is the feature set of phrase w. This probability is estimated from the training corpus using the Bayesian rule as given by

$$P(keyphrase|F_w) = P(keyphrase) \prod_{f \in F_w} P(f|keyphrase) \tag{13}$$

The class prior $P(keyphrase)$ is low, since the proportion of keyphrases to non-keyphrases in a document is very low. As a result of this imbalance between the classes, the probability $P(keyphrase|F_w)$ is low, and it is not possible to use strict thresholds for classification. For this reason, in contrast to other classification methods, thresholds are not applied for keyphraseness scores. When the target keyphrase size is 5, the top five ranking keyphrases are returned as the output, no matter how low the probability value is. Using this method, the prior probability can be neglected in calculations, as it will be the same for each candidate w.

Kea [7] reports a higher precision when the feature values are discretized using the minimum discrimination length (MDL) [34]. The features we have introduced behave similarly, and their precision decreases when supervised discretization is not applied to the features. Discretization is done by splitting the value ranges so as to minimize the entropy of the training population with respect to the probability of keyphraseness. For this reason, we apply MDL discretization to all the features.

4. Experiments and evaluation

4.1. Corpus

In this article, a corpus composed of 75 journal articles is used. The same corpus has previously been used in other keyphrase extraction research [6, 7, 14]. The corpus is composed of journal articles from different domains, as shown in Table 2. About 82% of the keyphrases appear in the articles, so 18% of the keyphrases cannot be extracted.

In order to highlight the disadvantages of the systems that depend solely on in-document features, an experiment using a corpus of shorter documents is conducted. To this end, the abstracts of the same journal articles are used. The average document length of the abstracts is 156 words, and 44.8% of the keyphrases appear in the abstracts – that is, 55.2% of keyphrases cannot be extracted.

Furthermore, not all of the keyphrases occur in the background corpus: 14.17% never appear in Wikipedia, and 16.6% appear in less than five different Wikipedia articles. It is possible to solve this problem by using a larger knowledge base such as a search engine, or a domain-specific corpus stored in a digital library. Also, in practical applications of extracted keyphrases, the importance of detecting such uncommon keyphrases is low.

4.2. Results

Keyphrase extraction systems are usually evaluated using precision and recall, which are defined in terms of sets of phrases. In a processed source document, let the set A be the author-assigned phrases, and let A' be the subset of A formed of phrases appearing in the document. Let E be the set of phrases extracted automatically by the system. The recall is calculated according to

$$Recall(A,E) = \frac{|A \cap E|}{|A'|} \tag{14}$$

Table 2. Corpus of journal articles and its attributes

Journal name	No. of articles	Keyphrases/document	Words/document
Journal of the International Academy of Hospitality Research	6	6.2	6,299
Psycholoquy	20	8.4	4,350
The Neuroscientist	2	6.0	7,476
Journal of Computer-aided Molecular Design	14	4.7	6,474
Behavioral & Brain Sciences Preprint Archive	33	8.4	17,522
All	75	7.5	10,781

Table 3. Keyphrase extraction full-text experiment results

Algorithms	Recall			Precision			KeyPerDoc		
	5	10	15	5	10	15	5	10	15
Kea	0.19	0.32	0.38	0.25	0.20	0.17	1.25	2.05	2.50
inDoc + QPPFeats	0.24	0.32	0.40	0.34	0.22	0.19	1.70	2.20	2.80
QPPFeats	0.11	0.21	0.28	0.16	0.15	0.13	0.80	1.45	1.95
inDoc	0.11	0.27	0.36	0.15	0.19	0.17	0.75	1.85	2.50

Table 4. Keyphrase extraction abstracts experiment results

Algorithms	Recall			Precision			KeyPerDoc		
	5	10	15	5	10	15	5	10	15
Kea	0.16	0.22	0.29	0.13	0.17	0.07	0.63	0.84	1.10
inDoc + QPPFeats	0.27	0.44	0.53	0.21	0.17	0.14	1.05	1.68	2.05
QPPFeats	0.29	0.47	0.55	0.22	0.18	0.14	1.11	1.79	2.11
inDoc	0.18	0.29	0.40	0.14	0.11	0.10	0.68	1.11	1.53

To avoid penalizing keyphrase extraction systems for keyphrases that cannot be extracted, the recall value is calculated with respect to the set A'. The precision is calculated from

$$Precision(A,E) = \frac{|A \cap E|}{|E|} \tag{15}$$

The test and training data are chosen so as to be compatible with the experiments performed by Turney [6] and Frank et al. [7]. Of the journal articles, 20 are reserved for testing and the remaining 55 documents are used for training. In the corpus of abstracts, 53 documents are used for training and 19 for testing; three have been omitted, as they lack an abstract.

Tables 3 and 4 present the results of the full-text and abstract experiments respectively. For both the experiments, the precision, recall and average number of correct keyphrases per document are given when 5, 10 and 15 keyphrases are extracted for each article. The results of Kea [7][5] are also provided for comparison. In Tables 3 and 4, *inDoc + QPP* denotes the experiments using all of the features defined in Table 1. The feature set *inDoc* denotes *tf*idf* and *firstPosition. QPPFeats* denotes all features, excluding those of *inDoc*.

In full-text articles, the Kea algorithm is able to extract 38% of the keyphrases appearing in the articles, when 15 keyphrases are extracted for each document. Journal articles concisely define the contribution of the document in early sections, and keyphrases are used more frequently in abstracts and introductory portions of the document. This is why the *firstPosition* feature achieves high accuracy in scientific articles.

As seen in Table 3, the effectiveness of *QPPFeats* is lower than that of Kea and *inDoc*. We have observed that QPP features tend to have similar values for domain-specific phrases, keyphrases and sub- or superphrases of keyphrases. In fact, for a superset of a keyphrase, a similar set of documents is usually retrieved from the background corpus. For example, for the keyphrase 'obsessive compulsive disorder' and the subphrases 'obsessive compulsive', 'compulsive disorder' as well as the superphrase 'obsessive compulsive personality disorder', similar sets of documents are retrieved from the background corpus. Since all QPP features are calculated using the retrieved documents, the feature values are almost identical to each other. In order to tackle this problem, and to improve the effectiveness of the system, we integrated QPP features with in-document features in full-text articles. As a result, the *inDoc + QPPFeats* system achieves the best recall values when compared with the *Kea* and *inDoc* algorithms.

The results of the experiment using abstracts, as shown in Table 4, reveal a defect of *inDoc* features in shorter documents. Whereas *tf*idf* and *firstPosition* are able to achieve high precision and recall values in full-text experiments, their performance is poor in the abstract corpus. As indicated previously, the *firstPosition* feature, depending on the structure of the document, is effective in full-text articles. However, in shorter documents, the *firstPosition*, which is the normalized distance from the start of the document to the word, is subject to more noise. Whereas a change in distance by a few words does not change the value of *firstPosition* in long documents, it changes the distance value significantly in short

documents. *tf*idf* values are formed of two components of term frequency and inverse document frequency. Inverse document frequency gets larger values when the phrase occurs in fewer documents. In short texts, most phrases occur once or a couple of times. Since frequencies of terms are similar, a high value of *tf*idf* is assigned to a phrase that appears infrequently in the corpus. Thus for short documents it is even possible to observe the highest *tf*idf* values in spelling errors and typos.

The QPP features are not extracted directly from the document, and can be calculated for any phrase, regardless of how many times it occurs in the text, if it ever does. This makes them more robust to changes in the length of the documents. For abstracts, the recall values of *QPPFeats + inDoc* and *QPPFeats* are better than those of the Kea algorithm. On the one hand, *QPPFeats + inDoc* correctly identifies 53% of author-assigned keyphrases appearing in the abstract when 15 keyphrases per article are extracted. On the other hand, it can be observed that in-document features degrade the effectiveness of *QPPFeats + inDoc* in short documents, since 55% of the keyphrases are identified when only *QPPFeats* are used. Both the *inDoc* and *Kea* algorithms are able to extract only a maximum of 40% of the keyphrases, which is 15% less than *QPPFeats* when 15 keyphrases are extracted. Their recall is about half of the QPP features when only five keyphrases are extracted.

One important advantage of *QPPFeats* is that it is possible to calculate them for phrases not appearing in the original document. In the keyphrase generation problem, in contrast to extraction, the algorithm should be able to generate phrases not appearing in the text, and should add keyphrase candidates from a prior knowledge, such as a background corpus or taxonomy. Expanding the extracted candidate keyphrases is a research topic by itself, and is left as a future work. However, in order to demonstrate the fact that *QPPFeats* can be used in keyphrase generation, we have performed an additional experiment. In the abstracts corpus we have manually added the 55% of the keyphrases that do not occur in their respective abstract to extracted candidate phrase lists, and repeated the experiment. In this setting, when the 15 top-scoring keyphrase candidates are selected, the number of correct keyphrases generated is improved by 42.5%, and the recall value is increased to 78%, with a precision of 20%. The precision value is even higher than the result of *inDoc + QPPFeats* in the full-text article experiments: that is, the *QPPFeats* can extract more keyphrases only by observing the abstracts.

When the QPP features are studied individually, it is observed that the two features $CosCentrTod_0$ and $avgCosTod_0$ have the greatest impact on keyphrase extraction. Our experiments suggest that using these two features provides the greatest improvement in keyphrase extraction. Two features – document perturbation (i.e. ranking robustness) and clarity – can be successfully used to improve the results in both QPP [21,22] and keyphrase extraction.

5. Conclusion

Most of the earlier work on keyphrase extraction focuses on research articles. However, there is an increasing interest in applying keyphrase extraction in shorter documents, such as Twitter messages [35] and news articles [12]. In this research, the potential problems of features commonly used in keyphrase extraction were shown through experiments. Although these features are useful in full-text articles, their effectiveness drops in short documents. Features extracted from a background corpus are able to solve this problem. We have shown that while the introduced QPP features improve keyphrase extraction in full-text articles, the improvement is much more significant for shorter documents like abstracts.

Furthermore, our features are not dependent on the occurrences of the phrases in the original document, and can be calculated for phrases that never appear in the document. All in all, this work aimed to establish a link between the problems of QPP and keyphrase extraction. We believe that this work contributes to the research on finding keyphrases by removing the constraints imposed by features directly extracted from the occurrences in the original document. A careful investigation of techniques for creating candidate keyphrase lists by mining related articles or semantically related phrases enables our algorithm to generate keyphrases. The techniques used in this article may lead to more general methods that are able to operate on different genres and perform generation instead of extraction.

Notes

1. Available at http://lucene.apache.org
2. The IDF component is used to weigh the effect of the terms, when the query is formed of multiple terms. Equation (1) differs from Jones [1], as it does not use the IDF component.
3. The dump file of 30 July 2010 retrieved from http://download.wikimedia.org is used.
4. Common English propositions and articles are excluded, and a stopword list of 452 words is employed.
5. Kea 3.0 version, downloaded from; http://www.nzdl.org/Kea/download.html

References

[1] Jones KS, Walker S, Robertson SE. A probabilistic model of information retrieval: development and comparative experiments. *Information Processing and Management* 2000; 36(6): 809–840.

[2] Gutwin C, Paynter G, Witten I, Nevill-Manning C, Frank E. Improving browsing in digital libraries with keyphrase indexes. *Journal of Decision Support Systems* 1999; 27(1–2): 81–104.

[3] Cronen-Townsend S, Zhou Y, Croft WB. Predicting query performance. In: *Proceedings of the 25th annual international ACM SIGIR conference on research and development in information retrieval (SIGIR)*, 2002. New York: ACM; 2002, pp. 299–306.

[4] Amati G, Carpineto C, Romano G. Query difficulty, robustness, and selective application of query expansion. In: *Advances in information retrieval*. Berlin: Springer, 2004, pp. 127–137.

[5] Yom-Tov E, Carmel D, Darlow A, Pelleg D, Errera-Yaakov S, Fine S. Juru at TREC 2005: query prediction in the terabyte and the robust tracks. In: Voorhees EM, Buckland LP (eds) *Proceedings of the 2005 text retrieval conference (TREC)*. NIST; 2005.

[6] Turney PD. Learning algorithms for keyphrase extraction. *Information Retrieval* 2000; 2(4): 303–336.

[7] Frank E, Paynter GW, Witten IH, Gutwin C, Nevill-Manning CG. Domain-specific keyphrase extraction. In: *Proceedings of the 16th international joint conference on artificial intelligence (IJCAI)*, 1999. San Francisco, CA: Morgan Kaufmann Publishers, pp. 668–673.

[8] Hulth A. Improved automatic keyword extraction given more linguistic knowledge. In: *Proceedings of the 2003 conference on empirical methods in natural language processing (EMNLP)*. Morristown: ACL; 2003, pp. 216–223.

[9] Barker K, Cornacchia N. Using noun phrase heads to extract document keyphrases. In: Hamilton HJ (ed.) *Proceedings of the 13th biennial conference of the Canadian Society on Computational Studies of Intelligence: Advances in artificial intelligence*. Berlin: Springer; 2000, pp. 40–52.

[10] Nguyen TD, Kan M-Y. Keyphrase extraction for scientific publications. In: *Asian digital libraries: Looking back 10 years and forging new frontiers*. Berlin: Springer; 2007, pp. 317–326.

[11] Mihalcea R, Tarau P. TextRank: bringing order into texts. In: *Proceedings of 2004 conference on empirical methods in natural language processing (EMNLP)*. Morristown: ACL; 2004, pp. 404–411.

[12] Wan X, Xiao J. Exploiting neighborhood knowledge for single document summarization and keyphrase extraction. *ACM Transactions on Information Systems* 2010; 28(2). doi: 10.1145/1740592.1740596.

[13] Turney PD. Coherent keyphrase extraction via web mining. In: *Proceedings of the 18th international joint conferences on artificial intelligence (IJCAI)*. San Francisco, CA: Morgan Kauffman Publishers, 2003, pp. 434–442.

[14] Ercan G, Cicekli I. Using lexical chains for keyword extraction. *Information Processing and Management* 2007; 43(6): 1705–1714.

[15] Fellbaum C (ed.). *WordNet: An electronic lexical database (language, speech, and communication)*. Cambridge, MA: MIT Press, 1998.

[16] Brin S, Page L. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems* 1998; 30(1–7): 107–117.

[17] Toutanova K, Klein D, Manning CD, Singer Y. Feature-rich part-of-speech tagging with a cyclic dependency network. In: *Proceedings of the 2003 conference of the North American chapter of the Association for Computational Linguistics on Human Language Technology*, 2003. Vol. 1, pp. 173–180.

[18] Manning CD, Raghavan P, Schütze H. *Introduction to information retrieval*. New York: Cambridge University Press; 2008.

[19] Cronen-Townsend S, Zhou Y, Croft WB. A framework for selective query expansion. In: *Proceedings of the 13th ACM international conference on information and knowledge management (CIKM)*. New York: ACM, 2004. pp. 236–237.

[20] He B, Ounis I. Query performance prediction. *Information Systems* 2006; 31(7): 585–594.

[21] Vinay V, Cox IJ, Milic-Frayling N, Wood KR. On ranking the effectiveness of searches. In: Efthimiadis EN, Dumais ST, Hawking D, Javelin K (eds) *Proceedings of the 29th annual international ACM SIGIR conference on research and development in information retrieval (SIGIR)*. New York: ACM, 2006. pp. 398–404.

[22] Zhou Y, Croft WB. Ranking robustness: a novel framework to predict query performance. In: Yu PS, Tsotras VJ, Fox EA, Liu B (eds) *Proceedings of the 15th ACM international conference on information and knowledge management (CIKM)*. New York: ACM, 2006, pp. 567–574.

[23] Porter MF. An algorithm for suffix stripping. *Program* 1980; 14(3): 130 – 137.

[24] Radev DR, Jing H, Sty M, Tam D. Centroid-based summarization of multiple documents. *Information Processing and Management* 2004; 40(6): 919–938.

[25] Han E-H, Karypis G. Centroid-based document classification: analysis and experimental results. In: Zighed DA, Komorowski HJ, Zytkow JM (eds) *Principles and practice of knowledge discovery in databases (PPKDD)*. Berlin: Springer, 2000, pp. 424–431.

[26] Kwok K-L, Grunfeld L, Dinstl N, Deng P. TREC 2005 robust track experiments using PIRCS. In: Voorhees EM, Buckland LP (eds) *Proceedings of the 2005 text retrieval conference (TREC)*. NIST; 2005.

[27] Hopkins B, Skellam JG. A new method for determining the type of distribution of plant individuals. *Annals of Botany* 1954; 18(2): 213–227.

[28] Cox TF, Lewis T. A conditioned distance ratio method for analyzing spatial patterns. *Biometrika* 1976; 63(3): 483–491.

[29] Ponte JM, Croft WB. A language modeling approach to information retrieval. In: *Proceedings of the 21st annual international ACM SIGIR conference on research and development in information retrieval (SIGIR)*. New York: ACM, 1998. pp. 275–281.
[30] Lavrenko V, Allan J, DeGuzman E, LaFlamme D, Pollard V, Thomas S. Relevance models for topic detection and tracking. In: *Proceedings of the 2nd international conference on human language technology research (HLT)*. San Francisco, CA: Morgan Kaufmann Publishers, 2002, pp. 115–121.
[31] Kullback S, Leibler RA. On information and sufficiency. *Annals of Mathematical Statistics* 1951; 22(1): 79–86.
[32] Bigi B. Using Kullback–Leibler distance for text categorization. In: Sebastiani F (ed.) *Proceedings of the European conference on information retrieval (ECIR)*. Berlin: Springer, 2003, pp. 305–319.
[33] Pinto D, Benedi J-M, Rosso P. Clustering narrow-domain short texts by using the Kullback–Leibler distance. In: Gelbukh AF (ed.) *Proceedings of the conference on computational linguistics and intelligent text processing (CICLING)*. Berlin: Springer, 2007, pp. 611–622.
[34] Fayyad UM, Irani KB. Multi-interval discretization of continuous-valued attributes for classification learning. In: *Proceedings of the international joint conference on artificial intelligence (IJCAI)*. San Francisco, CA: Morgan Kaufmann Publishers, 1993, pp. 1022–1029.
[35] Zhao WX, Jiang J, He J, Song Y, Achananuparp P, Lim EP, Li X. Topical keyphrase extraction from Twitter. In: *Proceedings of the 49th annual meeting of the Association for Computational Linguistics: Human language technologies*. Stroudsburg, PA: Association for Computational Linguistics, 2011, Vol. 1, pp. 379–388.