

Honours Year Project Report

Automatic Keyphrase Generation

By

Nguyen Thuy Dung

Department of Computer Science

School of Computing

National University of Singapore

2006/2007

Honours Year Project Report

Automatic Keyphrase Generation

By

Nguyen Thuy Dung

Department of Computer Science

School of Computing

National University of Singapore

2006/2007

Project No: H079250

Advisor: A/P Kan Min Yen

Deliverables:

Report: 1 Volume

Abstract

Keyphrases are useful for many purposes like document summarization, clustering and indexing. However, only a minority of documents have keyphrases assigned for them. As manually assigning keyphrases to documents can be expensive and time-consuming, automatic keyphrase generation has attracted attention of many researchers. In this project, we introduce a new feature set that illustrates the linguistics characteristics and the “logical position” of phrase in a given document. Empirical results show that the new feature set does have a significant improvement over the well-known Kea algorithm (Frank, Paynter, Witten, Gutwin, and Nevill-Manning, 1999). We have also created a corpus that has been annotated for keyphrases by multiple readers. We believe that this corpus can serve for future work on keyphrase generation.

Subject Descriptors:

H.3.1 Content Analysis and Indexing

Keywords:

keyphrase extraction, supervised machine learning

Implementation software and hardware:

Software: Java, Weka, Maxent, mxterminator, mxpost, umdhmm-v1.02

Acknowledgement

First and foremost, I would like to express my gratitude to Assistant Professor Min-Yen Kan for his support and guidance. Throughout the project, he has given me a lot of ideas and guided me patiently along the way.

I also thank my friends for their sharing, understanding and for their companion during the up and down of the project.

Above all, I thank my parents for being a constant support and encouragement. Though they cannot be here with me, they have always cared for me and given me strength.

Contents

Abstract	i
Acknowledgement	ii
1 Introduction	2
2 Literature Review	4
2.1 Keyphrase Assignment	4
2.2 Keyphrase Extraction	5
2.3 Other techniques	7
3 Methodology	9
3.1 Overview	9
3.2 Candidate phrases generation	10
3.2.1 Phrase identification	10
3.2.2 Acronym expansion	13
3.2.3 Case-folding and stemming	14
3.3 Feature set	14
3.3.1 Baseline feature set	14
3.3.2 Linguistics feature set	15
3.3.3 Section-related feature set	17
3.4 Keyphrase retrieval procedure	18
3.4.1 Model building	18
3.4.2 Keyphrase extraction	19
4 SectionFinder	21
4.1 Identifying sections	21
4.2 Associating headers with generic headers using Hidden Markov Model	22
4.3 Associating headers with generic headers using Maximum Entropy	24
4.4 Evaluation of two methods	24
5 Data Collection	26
5.1 Documents collection	26
5.2 Keyphrases collection	27
5.3 Final corpus	28
6 Empirical Results	29
6.1 Performance metric	29
6.2 Evaluation results	30
6.3 Analysis	31
7 Conclusions	33
A Suffixes of keyphrases	36

List of Figures

3.1	Input files pre-processing	10
3.2	System overview	11
3.3	Recovering noun phrase process	12
3.4	Part of speech tags' distribution of keyphrases	16
3.5	Suffixes' distribution of keyphrases	17
4.1	A representation of HMM as a graphical model. q_0, q_1, \dots, q_T are hidden states. y_0, \dots, y_T are observable outputs. π is the prior on the first state, and A is the transition probability matrix from one state to another. In this project, we assume homogeneous transition probability matrix, i.e. the transition matrix does not vary in time.	22

List of Tables

3.1	Linguistic feature set	17
3.2	Section-related feature set	18
4.1	List of generic headers	24
4.2	Accuracy of SectionFinder using HMM	24
4.3	Accuracy of SectionFinder using Maxent	25
5.1	Distribution of documents over number of keyphrase sets	28
6.1	Performance of the new feature set using Naïve Bayes and Maximum Entropy and Kea based on exact matching	30
6.2	t-test p-values measure significance against the baseline based on average number of exact matches	31
6.3	t-test p-values measure significance against the baseline based on average score	31
6.4	Sample output keyphrases	32

Chapter 1

Introduction

Keyphrases are defined as phrases that capture the main topics discussed in a document. As they offer a brief yet precise summary of a document content, they can be utilized for various applications. In an information retrieval (IR) environment, they serve as an indication of document relevance for users, as the list of keywords can quickly help determine whether a given document is relevant to their interest. As keyphrases reflect a document's main topics, they can be utilized to cluster documents into groups by measuring the overlap between the keyphrases assigned to them. Keyphrases also be used proactively in IR, in indexing. Here, good keyphrases supplement standard, full text indexing in assisting users in finding relevant documents.

Despite these known advantages of keyphrases, unfortunately, only a minority of documents have keyphrases assigned to them. This is because authors provide keyphrases only when they are instructed to do so (Frank et al., 1999). Besides, manually assigning keyphrases to documents is expensive and time-consuming. Therefore, there exists a great demand for automatic keyphrase generation.

In this project, we have proposed a new feature set that illustrates the *logical position* as well as the linguistics characteristics of a phrase in a given document. We have conducted evaluation of the new feature set using two different machine learning schemes. Performance of the two learning schemes are compared to published, well-known Kea algorithm for keyphrase extraction (Frank et al., 1999).

We have also created a corpus of more than 200 scientific publications, expressly assembled for keyphrase evaluation. Each publication was annotated by human readers in addition to keyphrases provided by the author. We believe that this corpus will be

useful in future work on keyphrase generation.

In the next chapter, we review previous approaches in automatic keyphrase generation. We then describe our overall methodology in Chapter 3, and present our logical position module implementation in Chapter 4: **SectionFinder**. Corpus collection is then described in detail in Chapter 5. Chapter 6 outlines our empirical evaluation of our system, discusses our results and concludes the thesis.

Chapter 2

Literature Review

Many past approaches have been proposed to solve the automatic keyphrase generation problem. They can basically be categorized into two main streams: keyphrase assignment and keyphrase extraction. This chapter provides a detailed description and comparison of these two research directions, summarizing their advantages and limitations. In the last section, we briefly discuss some techniques that are remotely related to keyphrase extraction.

2.1 Keyphrase Assignment

In keyphrase assignment, one first derives a list of potential keyphrases for a set of documents. Often, the potential keyphrase comes from a controlled vocabulary, perhaps conforming to a classification standard or standard set of subject headings, rather than from the document themselves. The problem is formulated as deciding whether a keyphrase is applicable to each individual document, using standard supervised machine learning. For each keyphrase, training documents that have been assigned the keyphrase are considered as positive examples, the rest are negative ones. A new document is then processed by all classifiers and mapped to all categories for which the classifiers determine it as a positive example.

Dumais, Platt, Hecherman, and Sahami (1998) proposed five learning methods for assigning categories to documents : Find Similar algorithm, Decision Trees, Naïve Bayes, Bayes Nets and Support Vector Machines. Each document is represented as a vector of words. For the Find Similar algorithm, a term weight is an average of its TFxIDF

weights in positive examples of the category. For the remaining learning schemes, for each category, k words are selected based on its mutual information with that category. Next, for a document, a binary feature is calculated for each of the k words. This feature shows whether that word occurs in the given document or not. Dumais et al. (1998) reported accuracy of about 90% for assigning 90 categories to 3000 documents using Support Vector Machines.

Pouliquen, Steinberger, and Ignat (2003) also used TFxIDF score in their keyphrase assignment method, but they used a larger list of potential keyphrases: Eurovoc thesaurus. Eurovoc consists of approximately 6000 index terms arranged into a hierarchy of eight levels using the relations Broader Term, Narrower Term and Related Term. They achieved a precision of 67% and recall of 63% based on human judgements for their top 8 terms.

The most important advantage of keyphrase assignment is that the assigned keyphrases are well-formed since they come from a controlled vocabulary. However, it requires human expertise to generate a list of potential keyphrases. Previous research results seem to indicate that a large training data is needed for the method to obtain satisfactory accuracy. For example, Pouliquen et al. (2003) used a corpus of 60,000 and Mark used 35,000 documents in their methods.

2.2 Keyphrase Extraction

In contrast, keyphrase extraction does not use a predefined list of potential keyphrases. Its purpose is to select the phrases from the document itself that best represent its topic. Keyphrase extraction approaches are realized in two stages: come up with a list of candidates, and select the most significant among them.

Barker and Cornacchia (2000) presented a simple system for choosing noun phrases from a document as keyphrases. A noun phrase is ranked based on three features: its length in words, its frequency and the frequency of its head noun.

Turney (1999) approached the keyphrase extraction problem as a supervised learning task. Turney (1999) did not restrict keyphrases to be noun phrases. All phrases of length from 1 to 3 consecutive words are treated as candidate phrases. For each candidate, a vector of nine feature values are computed. These nine features consist of 6 continuous

feature values : length in words, first position of stemmed phrase, first position of the earliest occurring stemmed word in stemmed phrase, frequency of stemmed phrase, frequency of the most frequent stemmed word in stemmed phrase, relative length of the most frequent whole phrase. The remaining 3 features are binary features that indicate whether the whole phrase is a proper noun, ends in a final adjective or contains a common verb. The main difference of this feature set from those features used by Barker and Cornacchia (2000) is that it takes into account the position of phrases. Finally, candidate phrases are ranked using decision tree induction. Turney (1999) also introduced a method based heuristic rules whose parameters are tuned by a genetic algorithm. On average, 80% of keyphrases extracted by Turney’s GenEx system were reported as acceptable for human readers

On the same direction, Frank et al. (1999) developed Kea, a key-phrase extraction system based on Naïve Bayes. Kea reduces the feature set to just two features: the TFxIDF score of the candidate phrase and the location of the phrase’s first occurrence. Despite this reduced feature set, Kea is reported to achieve the same performance level as GenEx. It is worth to notice that Kea’s performance can be improved significantly using an extra feature: keyphrase frequency, which measures how many times a phrase is assigned as a keyphrase by authors. Unfortunately, this feature is domain-specific, since though a given phrase is assigned as keyphrase for many documents in one discipline, it may not be good representative of documents in other disciplines. For example, “energy-momentum” can be a common keyphrase for documents in Physics area, but it can rarely occur as a keyphrase in Computer Science documents. For this reason, the training process must be repeated for each new domain. Another limitation is that it requires a large training set in order to achieve good performance (1000 training documents with author assigned keyphrases in comparison with 50 training documents in the original Kea).

Turney (2002) pointed out that generated keyphrases of prior keyphrases extraction algorithms were at times incoherent. That means some keyphrases may have no semantic relation with other keyphrases in the generated set. For example, in the keyphrase set $S = \{\text{test case, database, test suite, database testing, executed}\}$, *executed* seems to be an outlier as it does not fit well with other phrases. Therefore, he presented a new feature set called the coherence feature set to measure the coherence between candidate

phrases. This feature set is calculated using a two-pass method. In the first pass, candidate phrases are found using the original Kea system. In the second pass, L most probable phrases are chosen according to the estimated probabilities in the first pass. Then for each phrase in this set, a new feature set was calculated based on the statistical association between the candidate and a smaller group of K top candidate phrases ($K < L$). The statistical association is measured using pointwise mutual information and a web search engine. An advantage of this set of features is that it is not domain-specific as opposed to the keyphrase frequency feature in Kea. However, the computation required for the new feature set is very time-consuming because of the number of queries have to submit to a search engine. More specifically, it requires $2 + 2K$ queries to calculate a feature vector. Each query takes about one second, thus for a document with $K = 4$ and 100 feature vectors, it takes roughly 15 minutes to process one document.

Medelyan and Witten (2006) introduced an approach that can be seen as a combination of keyphrase assignment and keyphrase extraction. Though this approach also selects keyphrases from within the document, it exploits semantic information from a domain-specific thesaurus. Besides the features in the original Kea system, they are two additional features: length of a candidate phrase in words and the node degree. The node degree of a phrase is equal to the number of thesaurus links that connect it to other candidate phrases. The new features help improve the original Kea system from 12% in term of F-measure to 25.2%.

One advantage of keyphrase extraction is that it does not require human expertise to derive a list of potential keyphrases. However, while keyphrase assignment consistently selects the same phrase (or synonyms) for describing the same concept, keyphrase extraction loses this advantage. Another disadvantage of the extraction methods is that the extracted phrases might be ill-formed as the candidate selection process is not always accurate.

2.3 Other techniques

In this section, we describe other approaches which, in our opinion, are relevant to keyphrase generation. These approaches focus on generating keyphrases for a set of documents or terms for a particular topic. Furthermore, they require a large corpus and

hence may take a lot of time to run.

Kim and Wilbur (2001) showed three statistical techniques for identifying content bearing terms in a document. The first technique measures how strong a term t is related to its context. This is done by removing t from the subset of documents D_t that contain it to see if one can still distinguish D_t from the remaining documents. The second technique is based on a hypothesis that, non-content bearing terms should be distributed randomly among documents while content bearing terms tend to occur multiple times in a few documents. The third technique states that if phrase s occurs in document d allows us to predict that d is in dataset A as opposed to dataset B , then s is more likely a content bearing term in A .

Tomokiyo and Hurst (2003) proposed a language model approach for keyphrases extraction. According to them, good keyphrases must possess two features : phraseness and informativeness. Phraseness represents the extent to which a given word sequence is considered to be a phrase while informativeness refers to how well a phrase describes the key idea of a document. In their approach, they made use of two corpora: a foreground and a background corpus. The foreground corpus is the target document set from which keyphrases are extracted. The background corpus is the document set to which the target set is compared. They first built a unigram and a N-gram model for each corpus. Then, they used pointwise KL-divergence between these language models for scoring the phraseness and informativeness of a phrase. Finally, they unified these two scores into a single metric to rank extracted phrases.

A recent and separate model of keyphrase extraction is based on probabilistic topic models (Steyvers and Griffiths, 2005). The main idea of topic models is that a document is a mixture of topics and a topic is a probability distribution over words. Thus, topic models can be considered as generative models for documents, and dually, given a document one can infer the topic(s) responsible for generating that document. Topic models allow us to find out which terms are relevant to a particular topic.

Chapter 3

Methodology

3.1 Overview

In this project, we approached the keyphrase extraction problem as a classification task using supervised learning. First, a model is built from the training documents, each of which has a set of manually assigned keyphrases. A list of candidate phrases is generated for each document. Then for each candidate, we compute a set of feature values that consist of the baseline feature set from Kea (Frank et al., 1999) and additional features contributed by our work. Those candidates that were manually assigned as keyphrases are marked as positive examples and the remaining candidates are marked as negative examples. A model is then learned from the candidates' feature and class values using a suitable machine learning algorithm.

In the testing (or deployment) stage, we identify keyphrases for a new document, by first generating a list of candidates and computed feature values for each of them. Then the model is used to assign the probability for each candidate to be a keyphrase. Candidates with highest probabilities are selected into the final set of keyphrases. The number of output keyphrases can be specified by user. Figure 3.2 shows the overall description of our system.

In both the training and testing stage, the input files required for each document are files with following extensions:

1. SEC : a text file in which a document is divided into sections. The author assigned keyphrases are removed from this file.

2. POS : a text file which contains the corresponding part-of-speech tags of the texts in SEC file
3. KWD : a text file which contains manually assigned keyphrases.

However, since documents often found in PDF format, we use PDF files as raw input to our system. We need to pre-process these input files to retrieve the above required input files. Figure 3.1 shows the high level description of the pre-processing procedure.

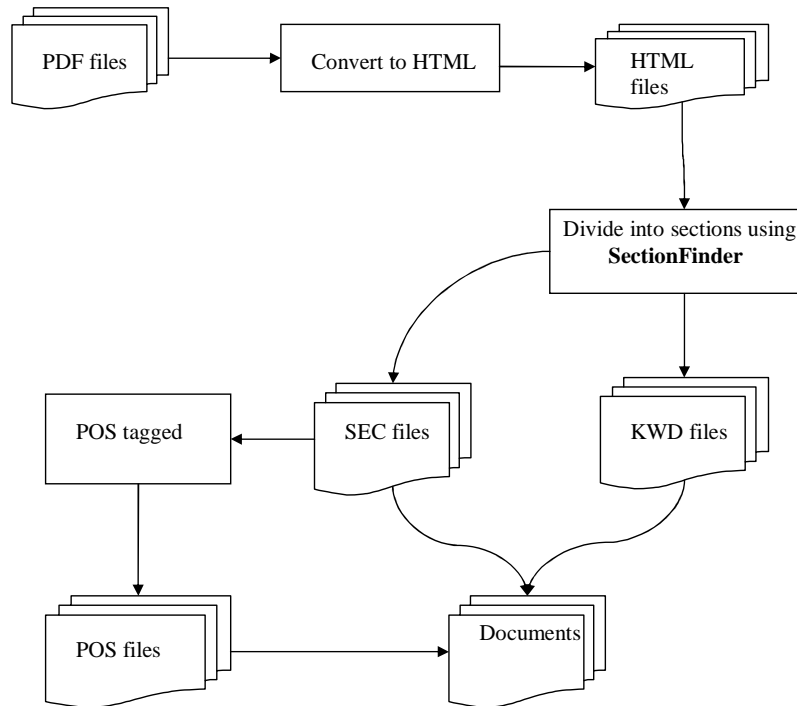


Figure 3.1: Input files pre-processing

Section 3.2 describes the candidates generation process. The feature set used in the system is elaborated in Section 3.3. Finally, in Section 3.4, we discuss the two machine learning schemes applied in our approach as well as post-processing operations of the whole keyphrase extraction procedure.

3.2 Candidate phrases generation

3.2.1 Phrase identification

Since noun phrases are often used to index documents' keyphrases, we only consider noun phrases as candidate phrases. Algorithm 1 shows how a list of candidates is generated:

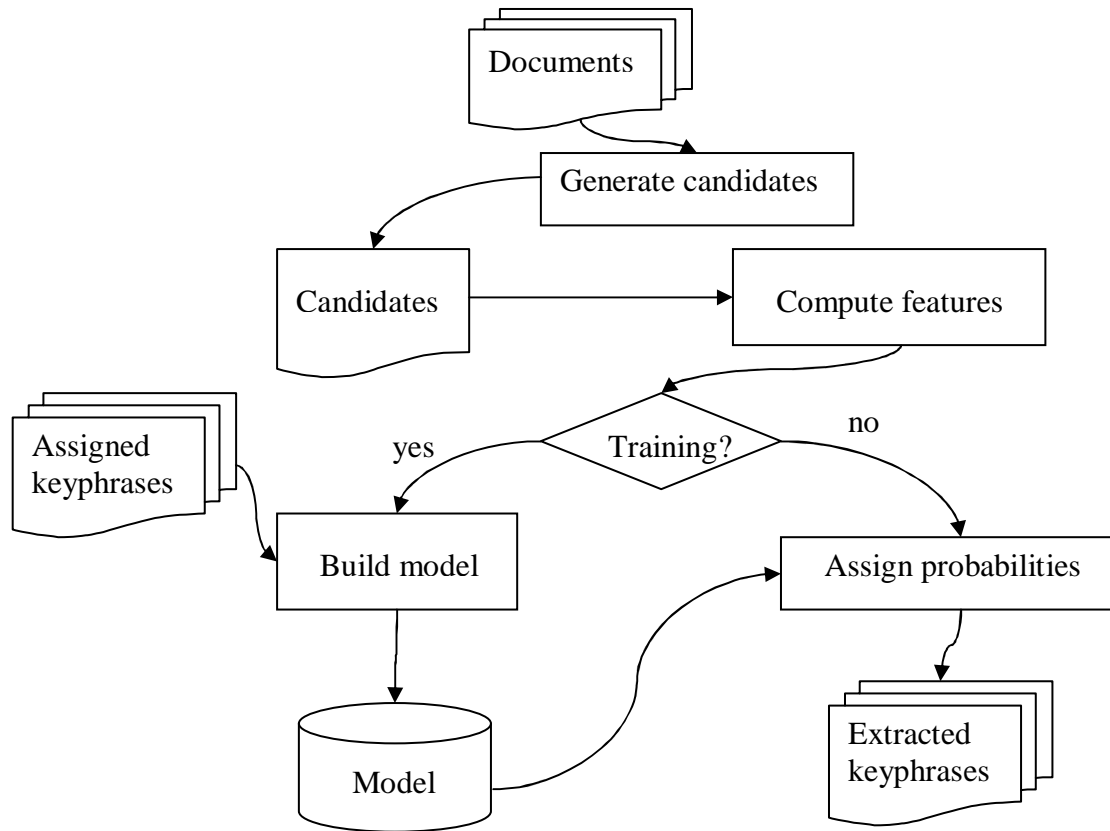


Figure 3.2: System overview

Algorithm 1 Algorithm for extracting noun phrases

- 1: Choose N most frequent head nouns $H_1 \dots H_N$ in document
 - 2: Recover all complete noun phrases for each $H_i \in H_1 \dots H_N$
 - 3: Select the noun phrases that appears at least twice in the document
-

Each document is first segmented into sentences using *mxterminator* and then part-of-speech tagged using *mxpost*¹. Head nouns are the rightmost words of the noun phrases with NN, NNS, NNP and NNPS tags. For example, the head noun in the phrase “machine learning techniques” is “techniques”. We do not consider noun phrases with post-modifiers like “algorithm that used in keyphrase extraction”. For each sentence in document, we identify all the head nouns which belong to N most frequent head nouns. Given a head noun, we move towards the left to retrieve the longest phrases with part-of-speech tags matched a predefined regular expression for noun phrases. The regular expressions used are:

- (NN |NNS |NNP |NNPS |JJ |JJR |JJS)*(NN|NNS|NNP|NNPS)
- (NN |NNS |NNP |NNPS)IN (NN |NNS |NNP |NNPS)

Figure 3.3 shows the noun phrase recovering process given a head noun in a sentence.

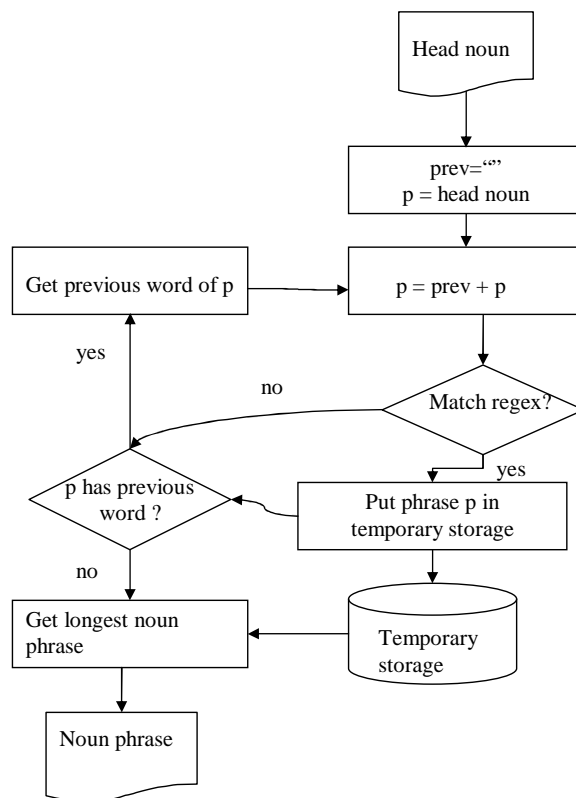


Figure 3.3: Recovering noun phrase process

¹<http://aye.comp.nus.edu.sg/portal/RPNLPIR/>

3.2.2 Acronym expansion

Authors often use acronyms for phrases that are repeated many times throughout the document. For example, to avoid repeating *Information Retrieval* many times, one could use *Information Retrieval* once and substitute *IR* thereafter. There is a need to link between a phrase and its acronym so that the system can know both the phrase and acronym refer to the same thing. To resolve this problem, we built a simple acronym detection program. Algorithm 2 shows how the acronym detector works. This detector can identify almost all acronyms with explicit definitions come right before or after them in the text. For example, in the sentence “we briefly discuss the notion of *Rendezvous Points (RPs)*”, there is an acronym “*RPs*” with its definition “*Rendezvous Points*”. However, this detector cannot resolve acronyms without explicit definitions in the text (common acronyms like “*IT*” often appear without being defined) as compared to the algorithm introduced by Nadeau and Turney (2005). Although algorithm 2 is simple, it performs quite well in matching the acronyms and their respective phrases (definitions) for documents in our corpus.

Algorithm 2 Algorithm for detecting acronym

```
1: Retrieve all the texts  $T_1 \dots T_N$  within parentheses () of document
2: for  $i = 1$  to  $N$  do
3:   if length of  $T_i < 2$  then
4:     Consider  $T_i$  neither being an acronym nor a definition, continue
5:   end if
6:   if ( $T_i$  is in uppercase or mixed-case) AND length of  $T_i < MAX$  then
7:     Assume  $T_i$  is an acronym
8:     Move toward the left to get its definition  $def_i$ 
9:     if  $def_i$  exists then
10:      Record the acronym  $T_i$  and its definition  $def_i$ 
11:    end if
12:   else
13:     Assume  $T_i$  is the definition
14:     Move toward the left to get its acronym  $acro_i$ 
15:     if  $acro_i$  exists then
16:      Record the acronym  $acro_i$  and its definition  $T_i$ 
17:    end if
18:   end if
19: end for
```

3.2.3 Case-folding and stemming

The final step in determining candidates is to case-fold all phrases and stem them using the iterated Lovins approach. Stemming and case-folding allow us to treat different variants of a phrase as one. For example *Bluetooth technology* and *Bluetooth technologies* are the same but without stemming they would be considered as different phrases. We kept the unstemmed phrase of each candidate, for displaying to the user in case it is actually a keyphrase. When there are more than one unstemmed phrase corresponded with one candidate, we chose the most frequent one. For example “voxel”, “voxelize” and “voxelization” are considered different variants of a candidate whose stemmed phrase is “voxel”. If “voxel” appears once, “voxelize” appears twice and “voxelization” appears five times in a document, then “voxelization” is kept as the unstemmed phrase of that candidate.

3.3 Feature set

One of the main goals of this project is to come up with new features which help to generate good keyphrases. We first experimented with two features used in Kea (Frank et al., 1999). Then, new features were incrementally integrated into our system. This process spanned through many iterations. In each iteration, after a new feature was added, evaluation of the new feature was conducted to see if it resulted in improvement over the baseline feature set. We also analyzed the output keyphrases to see what type of keyphrases which the new feature could not help to differentiate. Based on the data analysis, we tried to come up with other features to improve our keyphrase extraction algorithm. The new features added in addition to the baseline feature set can be divided into two subsets:

- Linguistics feature set
- Section-related feature set

3.3.1 Baseline feature set

The baseline feature set consists of two features used in Kea algorithm (Frank et al., 1999) : TFxIDF and distance.

TFxIDF

TFxIDF stands for term frequency times inverse document frequency, a widely used metric in information retrieval to determine the importance of terms. A term occurring frequently in the document but rarely in the rest of the collection is given high weight. Logarithm function is used in computing idf_i to dampen the effect relative to tf_{ij} .

Denote

- f_{ij} as frequency of term i in document j
- df_i = document frequency of term i = number of documents containing term i
- N = total number of documents

Term frequency of term i in document j

$$tf_{ij} = \frac{f_{ij}}{\max\{f_{ij}\}}$$

Inverse document frequency of term i :

$$idf_i = \log_2 \frac{N}{df_i}$$

TFxIDF weight of term i in document j :

$$w_{ij} = tf_{ij} \times idf_i$$

Distance

The distance of a phrase is calculated as the number of words that precede its first appearance, divided by the number of words in the document. This reflects the belief that keyphrases tend to appear at the beginning of the document.

3.3.2 Linguistics feature set

Jones and Paynter (2001) has shown that authors often choose good keyphrases for their documents. Therefore, we analyzed author keyphrases to know what characteristics a good keyphrase should possess. We focused on the linguistics characteristics of keyphrases assigned by authors.

Acronym

As stated in 3.2, authors often introduce acronyms for phrases that are used many times in a document. Intuitively, phrases that appear many times in a document are more relevant and may be a keyphrase. A binary feature indicates whether a candidate is an acronym is added to the feature set.

Part of speech

We observed that almost all of the author assigned keyphrases are noun phrases. This justifies why we only considered noun phrases as candidates in 3.2. Though the majority of author keyphrases are noun phrases, their part-of-speech tags come in different formats. Figure 3.4 shows the distribution of the part-of-speech tags of over 300 author keyphrases. The part-of-speech tag can be used as a feature to differentiate keyphrases from other candidates. For instance, a candidate has the unstemmed phrase “*incremental mining*” whose part-of-speech tag is “*JJ NN*”. We take the hash code of the part-of-speech tag as the value for this particular feature of this candidate.

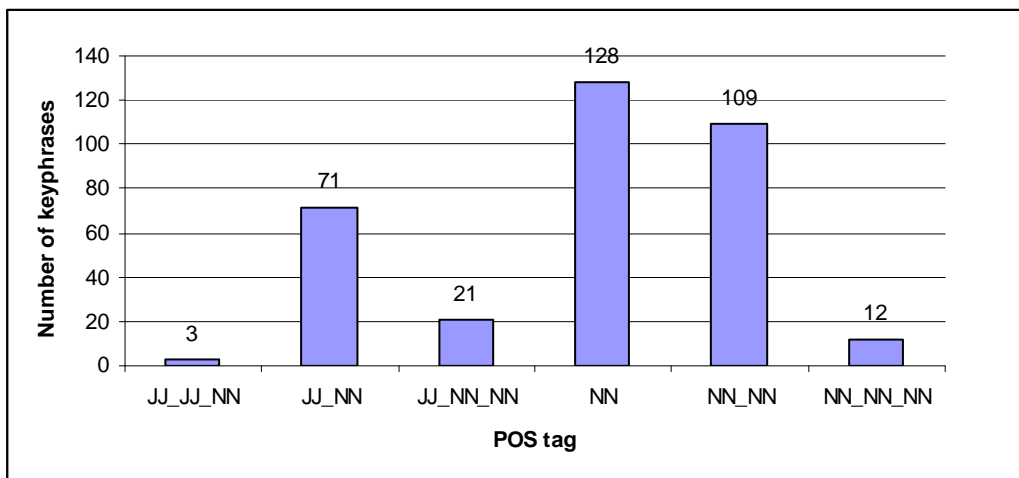


Figure 3.4: Part of speech tags’ distribution of keyphrases

Morphology

We also studied the morphology of keyphrases, in particular their suffixes. For each keyphrase, we define right hand side (RHS) to be its rightmost word and the rest as left hand side (LHS). We noticed that some suffixes such as *-ion*, *-ics*, *-ment* often appear on the RHS while others like *-ive*, *-al*, *-ic* appear on the LHS of a keyphrase. Figure 3.5

Name of feature	Description of feature	Value
acronym	is the candidate an acronym?	0,1
pos	hash code of part of speech tag of phrase	continuous
suf	hash code of concatenation of suffix of each word in phrases	continuous

Table 3.1: Linguistic feature set

illustrates the suffixes distribution of keyphrases (the entire data table is in Appendix A).

For each candidate, suffix of every word in its unstemmed phrase is retrieved. Then all the suffixes are put together in a string, each is separated by a space. Finally, the hash code of this string is used as a feature. Consider the example in the part of speech tag feature, the suffix string for that candidate is “al ing”, thus the value for this morphology feature is the hash code of “al ing”.

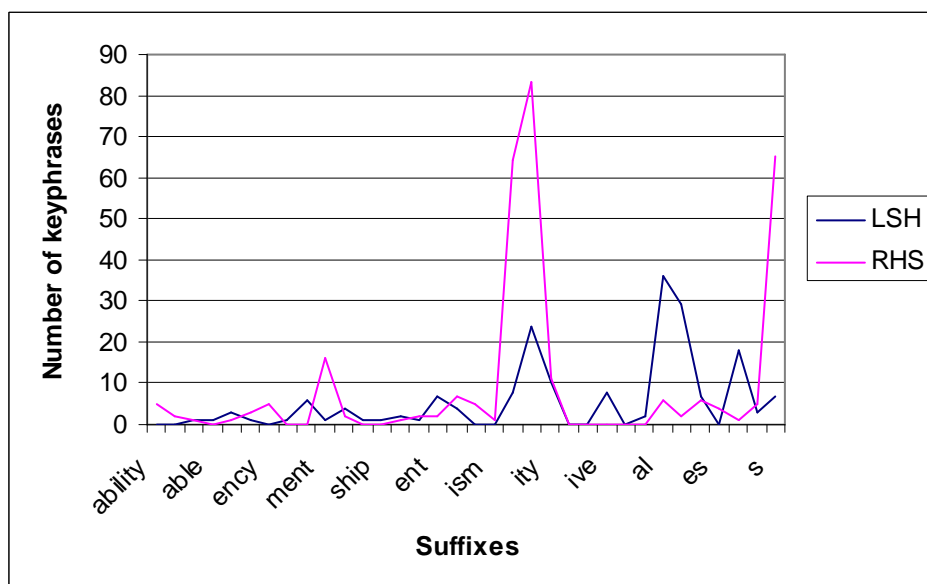


Figure 3.5: Suffixes’ distribution of keyphrases

3.3.3 Section-related feature set

Academic publications tend to follow a consistent structure, i.e. the majority in this group of publications contain the following generic sections: *abstract*, *introduction*, *related work*, *method*, *evaluation*, *conclusion* and *references*. We observe that different document sections may contribute different levels of relevance for its contained candidate phrases. More specifically, phrases that appeared in sections such as *abstract*,

Name of feature	Description of feature	Value
num_sec	number of sections in which phrase p appears — normalized by dividing by the total number of sections in document	continuous
a_freq	frequency of phrase p in <i>abstract</i> section	continuous
i_freq	frequency of phrase p in <i>introduction</i> section	continuous
c_freq	frequency of phrase p in <i>conclusion</i> section	continuous
r_freq	frequency of phrase p in <i>references</i> section — set to 0 if phrase p only appears in <i>references</i>	continuous
num_head	number of section headers in which phrase p appears	continuous
in_title	does phrase p appear in document title?	0,1

Table 3.2: Section-related feature set

introduction, *conclusion* of a given document were better representations of the subject of that document than phrases appeared in other sections. From this observation, we propose a new feature set that reflects the distribution of a phrase in the sections of a document. We called this the “logical position” of phrases.

In order to compute the new feature set, we need to identify to which generic section a phrase belongs. This is a non-trivial problem given the diversity of the section headers that actually occurs in documents. Thus, we built a subsystem called **SectionFinder** which performs this inference, which we consider a novel contribution of our thesis. In Chapter 4, we describe **SectionFinder** in detail.

3.4 Keyphrase retrieval procedure

3.4.1 Model building

Machine learning is commonly used to solve classification problems, in which the objective is to estimate a classification function $cl : X \rightarrow Y$ which maps an object $x \in X$ to its correct class $y \in Y$. Keyphrase extraction can be considered as a classification task in which X is a set of candidates and Y is predefined set of classes $\{keyphrase, non - keyphrase\}$. Thus, we used machine learning to identify keyphrase for documents. We have investigated two machine learning schemes: Naïve Bayes and Maximum Entropy.

- In the Naïve Bayes model, the classification function is implemented as the estimated conditional probability of x being in each class y . In our problem, for

each candidate, we only consider the probability of it being a keyphrase. Assume that all the features are independent given a class, the estimated probability of a candidate phrase with feature values $f_1 \dots f_n$ being a keyphrase is:

$$P[key|f_1, \dots, f_n] = \frac{\prod_{i=1}^n P[f_i|key] \times P[key]}{P[f_1, \dots, f_n]}$$

where $P[f_i|key]$ is the probability that feature i of a keyphrase has the value f_i , $P[key]$ is a priori probability that a phrase is a keyphrase and $P[f_1, \dots, f_n]$ is a normalization factor to make sure $P[key|f_1, \dots, f_n]$ is in $[0, 1]$. All these probabilities can be estimated by counting the number of corresponding events in the training data.

- **Maximum Entropy** : Maximum Entropy (ME) is a probability distribution modeling technique. The principle of ME modeling is to model all that is known and assume nothing about which is unknown. In other words, given a training dataset, choose a model consistent with all the constraints, but otherwise as uniform as possible. In ME, the classification function is implemented with a conditional probability model p by choosing the class with the highest conditional probability:

$$cl(x) = \arg \max_y p(y|x)$$

The probability model p is that which maximizes the uncertainty (hence the name Maximum Entropy) so as to get a model assumes nothing about which is unknown.

Some features in our feature set are continuous. According to Frank et al. (1999) Naïve Bayes learning method performs better when we discretize the feature values. We also need to discretize the feature values prior to apply the Maximum Entropy learning scheme, since Maximum Entropy treats all the features as strings. We made use of the discretization scheme in Weka.

3.4.2 Keyphrase extraction

To generate keyphrases for a new document, our system first determines the candidate phrases and their feature values, Naïve Bayes and Maxent are then applied to predict the probability whether each candidate is a keyphrase. The candidate phrases are ranked

according to this probability.

Finally, post processing operations are applied to select the most significant phrases. In this step, if two phrases have the same probability, TFXIDF score is used as a tie-breaker. We also remove phrases that is a sub-phrase of a higher ranking phrase. We return the first r phrase to users, r is input by users.

Chapter 4

SectionFinder

Although academic publications share a consistent structure, their actual section headers may not be the same. A methodology part does not always have the header "Methodology", furthermore it may correspond to more than one section in a document. Hence, the **SectionFinder**'s main function is to automatically associate each section with its corresponding generic section. In implementing **SectionFinder**, we tried two different methods: Hidden Markov Models (HMM) and Maximum Entropy. We applied Maximum Entropy in our final implementation of **SectionFinder** since it achieved a higher performance than using an HMM. In the next sections, we will elaborate the implementation of **SectionFinder** and the evaluation of the two methods.

4.1 Identifying sections

The first step in building **SectionFinder** was to locate the position of the sections in a document. To achieve this, we made use of the font characteristic of the text in a given document. This information is extracted from the STYLE block of the document's HTML file.

```
<STYLE type="text/css">
<!--
    .ft0{font-size:25px;font-family:Times;color:#000000;}
    .ft1{font-size:16px;font-family:Times;color:#000000;}
    .ft2{font-size:13px;font-family:Times;color:#000000;}
    .ft3{font-size:16px;font-family:Times;color:#000000;}
-->
</STYLE>
```

We assumed that every section headers of a document have the same font, i.e. same font-size and font-family. Moreover, the font of the section headers must be different from the font of the section content. With these two assumptions, we first acquired the font information of the *abstract* header. Then, we iterated through the document, identified the actual header of every section based on the following rules

1. Header’s font must be the same as the *abstract* header’s.
2. Header must start with the current section number.
3. Header must have the same orthography with other headers (except for the *abstract*, *acknowledgement* and *references* header)

By knowing the location of the section headers, we could retrieve other information about each section: section number and section content. Next, we went on to automatically map the section headers with the generic headers.

4.2 Associating headers with generic headers using Hidden Markov Model

A hidden Markov model (HMM) is a graphical model that is often used for modeling sequential data in which samples are not i.i.d but rather appear in certain order. Figure 4.1 shows the graphical model representation of HMM. Under HMM, the system can

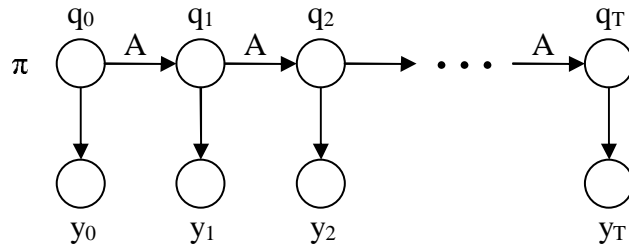


Figure 4.1: A representation of HMM as a graphical model. q_0, q_1, \dots, q_T are hidden states. y_0, \dots, y_T are observable outputs. π is the prior on the first state, and A is the transition probability matrix from one state to another. In this project, we assume homogeneous transition probability matrix, i.e. the transition matrix does not vary in time.

be in some pre-known states, and can switch from one state to another. The current state of the system, however, is assumed to be “hidden”. Instead, the system emits some output that is observable. This output is assumed to be dependent only on system

current state. Furthermore, HMM assumes the Markov property that the current state is dependent only on its immediate previous state. All these assumptions are captured in HMM graphical model. In short, the joint probability distribution $p(q, y)$ factorizes as follows

$$p(q, y) = \underbrace{p(q_0)}_{\pi} \prod_{t=1}^T p(q_t|q_{t-1}) \prod_{t=0}^T p(y_t|q_t)$$

Training HMM is to estimate the prior $\pi = p(q_0)$ and the transition probability $p(q_t|q_{t-1})$, given that both hidden states and outputs are observed. Once the prior and transition probabilities have been learned, HMM can then be used to estimate the most probable sequence of hidden states, given the sequence of outputs, as well as estimate the probability that the system emits such a sequence of outputs. All these can be done using classical forward-backward or junction-tree algorithms.

HMM is particularly suitable for our problem, as the sections do not appear independently, but rather in a particular flow of the document. The introduction part is likely to be followed by the methodology section rather than the conclusions. In this scenario, the hidden states are the general section headers we want to predict, and the observed outputs are the actual headers found in the documents. The set of possible states is then the set of possible generic headers, which we choose to be the top 14 most frequent headers in our corpus, as shown in Table 4.1

In this project, we do not train HMM on data but compute the probabilities directly. The transition probability from state i to state j is the probability of header i coming before header j . The output probability $p(y_t|q_t)$ is the probability that header H'_k is used for a section with generic header H_i , and is calculated using Jaccard similarity coefficient

$$J(H_i, H'_k) = 0.5 \times \frac{|H_i \cap H'_k|}{|H_i \cup H'_k|} + 0.5 \times (1 - |\text{Rel}(H_i) - \text{Rel}(H'_k)|)$$

- $|H_i \cap H'_k|$ = the number of common words between H_i and H'_k
- $|H_i \cup H'_k|$ = the number of words in H_i and H'_k
- $\text{Rel}(H_i)$ = the relative position of $H_i = \frac{\text{number of } H_i}{\text{Total number of sections in document}}$

And lastly, the probability that the initial state is the probability of header i being

the first header of the document. With these probabilities, we use HMM to derive the generic section headers.

abstract	categories and subject descriptors	general terms
introduction	background	method
conclusions	references	evaluation
related works	acknowledgments	applications
motivation	implementation	

Table 4.1: List of generic headers

4.3 Associating headers with generic headers using Maximum Entropy

Since Maximum Entropy is a powerful method for constructing statistical models of classification tasks, we tried to apply it to our section identification problem. We manually assigned the generic header from the headers in Table 4.1 for every section of the training documents. Then, we used Maximum Entropy model to assign header for every section in the testing documents. We used the following feature set in the Maximum Entropy model: section number, section relative position, previous section header, current section header.

4.4 Evaluation of two methods

To assess the performance of both approaches, we used a data set of 100 documents. Sections in each document are annotated with suitable generic headers. There were 1020 section headers in total.

For the untrained HMM, we read through the results generated by HMM, then we reported how many headers were correctly assigned.

# headers	# correctly assigned headers	Accuracy
1020	369	0.36

Table 4.2: Accuracy of **SectionFinder** using HMM

For the maximum entropy-based approach, we split the data set into 10 folds. For each of the n iterations, the n^{th} fold was taken to be the test set, and the rest $n - 1$ folds were used as the training set. The accuracy of the **SectionFinder** using Maximum

Entropy is shown in 4.3.

# headers	# correctly assigned headers	Accuracy
1020	938	0.92

Table 4.3: Accuracy of **SectionFinder** using Maxent

Chapter 5

Data Collection

Evaluating keyphrases has shown to be subjective and difficult in many previous works. According to Medelyan and Witten (2006) the evaluation of keyphrase extraction algorithms requires multiple judgments and cannot rely on a single set of keyphrases provided by a paper’s author. We followed this line of argument and sought to construct a data set in which each document contains multiple annotations for keyphrases. To do this, we first had to collect documents and then collect keyphrases. We show how this was done in section 5.1 and 5.2.

5.1 Documents collection

Documents with author assigned keyphrases are needed in both training and testing phase of our method. To retrieve documents that have keyphrases assigned to them, we used **googleFetch**¹ (which uses the Google SOAP API) with the following queries :

- `googleFetch -Q “keywords general terms permission acm filetype pdf” -n 200`
- `googleFetch -Q “keywords general terms permission filetype pdf” -n 500`

These documents were converted from PDF to HTML using **pdfEdit995**². We have also tried other tools in conversion from PDF to HTML such as : **PDF2HTML v1.6**, **PDF Ripper**. We chose **pdfEdit995** since it worked well with documents in both 1-column and 2-columns format. The documents were then converted from HTML to plain text. We did not convert directly from PDF to plain text because of two reasons.

¹http://aye.comp.nus.edu.sg/portal/RPNLPIR/google_web_api.html

²<http://www.pdfedit995.com/>

The first reason is that conversion tools from PDF to plain text at times do not work properly with documents in 2-columns format. More importantly, we want to make use of the *font information* in the HTML files for our **SectionFinder** module. Afterwards, we eliminated all duplicated documents based on each document’s size and content. We also removed documents that do not belong to Computer Science area or do not contain author assigned keyphrases.

With the help of **SectionFinder**, each document is broken down into sections. These are stored in a separated XML file. This enables partial retrieval of documents in case there is a need for a particular section e.g. abstract or introduction.

Using this method, we obtain a corpus consisting of 211 documents, each has 4 different formats: PDF, HTML, plain text, and XML

5.2 Keyphrases collection

Although author assigned keyphrases are usually viewed as a good representation of the subject of a document, they may not be able to cover all the good keyphrases in a document. This is because assigning keyphrases for documents is a subjective task, keyphrases assigned by one annotator are not the only “correct” ones. Because of this observation, we have conducted an experiment which asks human readers to provide keyphrases for documents.

The experiment was done via email. We first emailed students and staff in School of Computing of National University of Singapore(NUS) to invite them to participate in the experiment. To ensure that all the participants are familiar with reading Computer Science papers, we have limited the students to be third and fourth year students of School of Computing in NUS.

More than 80 subjects responded to our invitation. We divided these subjects into groups of 4. In the next step, links to three PDF papers were allocated to each of the subjects through email. In these three papers, there was one paper that had been annotated by us. This annotated paper was meant to measure how carefully the subjects considered the tasks. Subjects in the same group are allocated the same annotated paper. The papers have the minimum length of 4 pages and the maximum length of 12 pages. The author’s keyphrases were removed so as not to influence annotators’ decision. Each

# documents	# keyphrase sets
55	1
135	2
14	3
5	4
2	5

Table 5.1: Distribution of documents over number of keyphrase sets

subject was asked to first read each paper for 10 minutes. Then, they were asked to come up with a list of 10 keyphrases for each paper.

However, only 40 subjects returned back the results. Including documents that were annotated by us, there are 156 annotated documents in total. Table 5.1 shows the distribution of documents over the number of keyphrase sets. The number of keyphrase sets also includes the author assigned keyphrases.

5.3 Final corpus

To our knowledge, no corpus of articles exists that has been annotated for keyphrases by multiple subjects. A significant contribution of this thesis is to assemble a corpus of more than 200 scientific publications for keyphrase evaluation. Each document may contain more than one set of keyphrases: one from author and the remaining sets from other annotators. The corpus is available at <http://wing.comp.nus.edu.sg/downloads/keyphraseCorpus>. Users may find links to PDF, HTML, plain text, and XML files of each document. Extra information such as title, categories, subject descriptors, etc., if needed, can easily be extracted from XML format.

Chapter 6

Empirical Results

6.1 Performance metric

There are two approaches to evaluate keyphrase extraction algorithms. The first approach involves human evaluation of generated keyphrases. In this approach, subjects are provided with a document and a phrase list and asked to rank the relevance of each phrases to the given document. The main disadvantage of this approach is that it requires human expertise. The second approach adopts the standard Information Retrieval metrics of precision and recall to measure how well the generated keyphrases match the author assigned keyphrases. However, there are also many problems with evaluation based on author assigned keyphrases (Barker and Cornacchia, 2000). First, author keyphrases do not always appear in the text of the document to which they belong. Also, authors might choose keyphrases for purposes other than document description, or the number of keyphrases provided is far fewer than may be extracted automatically. And lastly, only a small number of documents have assigned keyphrases. Despite of these shortcomings, the study of Jones and Paynter (2001) proved that author keyphrases are good representations of the subject of a document.

We followed the second approach in assessing the performance of our keyphrase extraction method. In order words, we measured the extent to which the generated keyphrases match the manually assigned keyphrases. However, assigning keyphrases is a subjective task, each annotator may assign different keyphrases for the same document. Therefore, we proposed an evaluating scheme that takes into account the degree of agreement between annotators on their annotated keyphrases. For our data set, 120

	Average # of exact matches	Average score based on weight
Baseline Kea	3.03	3.61
Naïve Bayes	3.25	3.84
Maximum Entropy	3.03	3.58

Table 6.1: Performance of the new feature set using Naïve Bayes and Maximum Entropy and Kea based on exact matching

documents are selected, each of which has exactly 2 keyphrase sets : one is assigned by author and the other is assigned by human reader. These two sets are combined into one standard keyphrase set.

For each document, the accuracy is the number of matches among keyphrases in the standard set and ten top-ranking phrases output by keyphrase extraction algorithms. All the accuracies reported in this chapter are obtained using 10-fold cross validation.

6.2 Evaluation results

We first chose to assess the performance by counting the exact matches between the generated keyphrases and the manually assigned keyphrases. A key difference in our assessing method is that keyphrases are given different weights based on how many original keyphrase sets in which they appear. Let N be the number of keyphrases set in which a phrase p appears, then its weight $w(p)$ is computed as

$$w(p) = 1 + \ln N$$

In order to evaluate the effect of introducing our new feature set, we compare the performance of our system against Kea system (Frank et al., 1999). Table 6.1 shows the performance of our systems using 2 different machine learning algorithms, as well as the performance of Kea system. We perform 2-tail paired t-test to see whether the improvements are significant. The corresponding p-values are shown in Table 6.2 and 6.3.

The empirical results indicate that using Naïve Bayes classification method, the new feature set does have a significant improvement over the baseline. However, applying the new feature set in Maximum Entropy model has similar results in comparison with the baseline.

	Average # of exact matches	t-test p-value
Baseline Kea	3.03	
Naïve Bayes	3.25	0.024
Maximum Entropy	3.03	0.5

Table 6.2: t-test p-values measure significance against the baseline based on average number of exact matches

	Average score based on weight	t-test p-value
Baseline Kea	3.61	
Naïve Bayes	3.84	0.033
Maximum Entropy	3.58	0.412

Table 6.3: t-test p-values measure significance against the baseline based on average score

6.3 Analysis

In section 6.2, it has been shown that the new feature set does have an improvement over the baseline. However, both the baseline and our algorithms occasionally generate bad keyphrases. Table 6.4 shows the sample output for a given document. Output keyphrases that match with assigned keyphrases are presented in italic font. This document can be found at <http://wing.comp.nus.edu.sg/downloads/keyphraseCorpus/36/36.pdf>. Phrases such as “data” and “cell” are too general to be keyphrases. These phrases appear many times in the document, thus they have high TFxIDF scores. In addition, they also appear in important sections like abstract, introduction which results in their section-related features are the same with those of correct keyphrases. This explains why they are returned by the classifiers as keyphrases.

Another observation is that long phrases (phrases with three words or more) are rarely generated as keyphrases. As shown in Table 6.4, out of 30 generated keyphrases, only one phrase with length of three words are returned.

Assigned keyphrases	Kea	Naïve Bayes	Maximum Entropy
Neural Network Algorithm 3G network Visualization Capability Cluster Analysis Self Organizing Map hierarchical clustering Key Performance Indicator of Handover Two-Phase Clustering Algorithm soft handover Histograms Decrease in Computational Complexity mobility management data mining neural networks	handover <i>soft handover</i> 3G clusters <i>3G network</i> cell cell pairs SHO active set handover measurements	clusters <i>soft handover</i> data <i>3G network</i> interesting clusters handover attempts method <i>neural network</i> measurements handover measurement	<i>soft handover</i> clusters SHO <i>3G network</i> mobile station <i>histograms</i> cell <i>Self Organizing Map</i> interesting clusters active set

Table 6.4: Sample output keyphrases

Chapter 7

Conclusions

In this project, we have presented a new feature set that reflects the “logical position” of phrase in a given document. Two machine learning schemes are used to evaluate the new feature set : Naïve Bayes and Maximum Entropy model. Applying the new features in Naïve Bayes model does have a significant improvement against the baseline. However , it has similar results with the baseline Kea (Frank et al., 1999) when applied in Maximum Entropy model.

We have also collected a corpus of more than 200 scientific publications. Each publication was annotated by human readers for additional keyphrase set beside the set provided by author. We believe that this corpus will be useful in future work on keyphrase extraction.

Although the new feature set has resulted in a significant improvement over the baseline, there are quite many issues that are still open which may be solved in the future. One of the main issues is that general words are occasionally returned as keyphrases. Also, long phrases are rarely returned as keyphrases although they are in fact good keyphrases.

In the future, we intend to use *WordNet* to rank phrases based on their generalities in order to solve the problem with general keyphrases. And we also want to investigate the collocations of words so as to come up with new features that favor long phrases.

Bibliography

- Barker, K., and Cornacchia, N. (2000) Using noun phrase heads to extract document keyphrases. In *AI '00: Proceedings of the 13th Biennial Conference of the Canadian Society on Computational Studies of Intelligence*, pages 40–52, London, UK, 2000. Springer-Verlag. ISBN 3-540-67557-4.
- Dumais, S. T., Platt, J., Hecherman, D., and Sahami, M. (1998) Inductive learning algorithms and representations for text categorization. In *Proc. 7th International Conference on Information and Knowledge Management CIKM*, pages 148–155, 1998.
- Frank, E., Paynter, G. W., Witten, I. H., Gutwin, C., and Nevill-Manning, C. G. (1999) Domain-specific keyphrase extraction. In *IJCAI*, pages 668–673, 1999.
- Jones, S., and Paynter, G. W. (2001) Human evaluation of kea, an automatic keyphrasing system. In *ACM/IEEE Joint Conference on Digital Libraries*, pages 148–156, 2001. URL citeseer.ist.psu.edu/jones01human.html.
- Kim, W., and Wilbur, W. J. (2001) Corpus-based statistical screening for content-bearing terms. *J. Am. Soc. Inf. Sci. Technol.*, 52(3):247–259, 2001. ISSN 1532-2882. doi: [http://dx.doi.org/10.1002/1097-4571\(2000\)9999:9999<::AID-ASI1588>3.3.CO;2-Z](http://dx.doi.org/10.1002/1097-4571(2000)9999:9999<::AID-ASI1588>3.3.CO;2-Z).
- Mark, K. E. Interlingual indexing across different languages. URL citeseer.ist.psu.edu/727996.html.
- Medelyan, O., and Witten, I. H. (2006) Thesaurus based automatic keyphrase indexing. In *JCDL '06: Proceedings of the 6th ACM/IEEE-CS joint conference on Digital libraries*, pages 296–297, New York, NY, USA, 2006. ACM Press. ISBN 1-59593-354-9. doi: <http://doi.acm.org/10.1145/1141753.1141819>.

- Nadeau, D., and Turney, P. D. (2005) A supervised learning approach to acronym identification. In *Canadian Conference on AI*, pages 319–329, 2005.
- Pouliquen, B., Steinberger, R., and Ignat, C. (2003) Automatic annotation of multilingual text collections with a conceptual thesaurus, 2003.
- Steyvers, M., and Griffiths, T. (2005) Probabilistic topic models. In Landauer, T., Mcnamara, D., Dennis, S., and Kintsch, W., editors, *Latent Semantic Analysis: A Road to Meaning*. Laurence Erlbaum, 2005.
- Tomokiyo, T., and Hurst, M. (2003) A language model approach to keyphrase extraction. In *Proceedings of ACL Workshop on Multiword Expressions*, 2003.
- Turney, P. (1999) Learning to extract keyphrases from text, 1999.
- Turney, P. (2002) Coherent keyphrase extraction via web mining, 2002.

Appendix A

Suffixes of keyphrases

Table A shows the suffixes distribution of 344 author assigned keyphrases.

suffix	LHS	RHS	suffix	LHS	RHS
-ability	0	5	-metry	0	2
-ology	1	1	-able	1	0
-ance	3	1	-ence	1	3
-ency	0	5	-ible	1	0
-ical	6	0	-ment	1	16
-ness	4	2	-less	1	0
-ship	1	0	-ant	2	1
-ate	1	2	-ent	7	2
-ics	4	7	-ies	0	5
-ism	0	1	-ing	8	64
-ion	24	83	-ity	10	11
-ive	8	0	-ous	2	0
-al	36	6	-ed	29	2
-er	7	6	-es	0	4
-ic	18	1	-or	3	5
-s	7	65			