International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:1, No:9, 2007

# Extraction of Significant Phrases from Text

Yuan J. Lui

***Abstract*—**Prospective readers can quickly determine whether a document is relevant to their information need if the significant phrases (or *keyphrases*) in this document are provided. Although keyphrases are useful, not many documents have keyphrases assigned to them, and manually assigning keyphrases to existing documents is costly. Therefore, there is a need for automatic keyphrase extraction. This paper introduces a new domain independent keyphrase extraction algorithm. The algorithm approaches the problem of keyphrase extraction as a classification task, and uses a combination of statistical and computational linguistics techniques, a new set of attributes, and a new machine learning method to distinguish keyphrases from non-keyphrases. The experiments indicate that this algorithm performs better than other keyphrase extraction tools and that it significantly outperforms Microsoft Word 2000's AutoSummarize feature. The domain independence of this algorithm has also been confirmed in our experiments.

***Keywords*—**classification, keyphrase extraction, machine learning, summarization

## I. Introduction

With the proliferation of the Internet and the huge numbers of documents it contains, the provision of summaries of these documents has become more and more important ('document' is regarded as being synonymous with 'text' in this paper). Prospective readers can quickly determine whether a document is relevant to their information need if the significant phrases (or keyphrases) in this document are provided. Keyphrases give a short summary of the document and provide supplementary information for the readers, in addition to titles and abstracts. Even though keyphrases are useful, only a small minority of documents have keyphrases assigned to them, and manually assigning keyphrases to existing documents is very costly. Therefore, there is a need for automatic keyphrase extraction [4]-[6], [15]-[17].

Automatic keyphrase extraction is the identification of the most important phrases within the body of a document by computers rather than human beings. It normally involves the use of statistical information. There is no controlled vocabulary list, so, in theory, any phrase within the body of the document can be identified as a keyphrase. When authors assign keyphrases without a controlled vocabulary list, typically 70-90% of their keyphrases appear somewhere in their documents [17]. Keyphrases are similar to keywords, except that the document is summarized by a set of phrases rather than words.

Keyphrase extraction is a classification task: a document can be seen as a set of phrases, and a keyphrase extraction algorithm should correctly classify a phrase as a keyphrase or a non-keyphrase. Machine learning techniques can automate this task if they are provided with a set of training data composed of both keyphrase examples and non-keyphrase examples. The data are used to train the algorithm to distinguish keyphrases from non-keyphrases. The resulting algorithm can then be applied to new documents for keyphrase extraction. Previous work shows that the training data and the new documents need not be from the same domain, though the performance of the algorithm can be boosted significantly if they are [4].

This paper introduces a new domain independent keyphrase extraction algorithm called *KE*. KE is not tied to a specific domain; it is designed to summarize a given document, which can be on any topic (excluding poetry and other similar works of literature), in a few keyphrases automatically extracted from the body of that document. Unlike other keyphrase extraction algorithms, KE uses a combination of statistical and computational linguistics techniques, a different set of attributes, and a different machine learning method to extract keyphrases from documents. The experiments indicate that KE performs better than other keyphrase extraction tools and that it significantly outperforms Microsoft Word 2000's AutoSummarize feature. The domain independence of this algorithm has also been confirmed in our experiments.

Section II summarizes related work by other researchers. Section III introduces the KE algorithm and compares it with other keyphrase extraction algorithms. The experimental results are presented in Section IV. Section V discusses the results. Section VI concludes this paper and discusses future work.

## II. Related work

This section discusses two important term weights (i.e. term frequency and inverse document frequency), two important keyphrase extraction algorithms (i.e. GenEx and Kea) and some recent ones. Though GenEx and Kea were introduced in the late 1990s, they remain rather important and are reviewed in most of the papers on keyphrase extraction.

### *A. TF×IDF*

The vector space model [11] suggests that a document (or query) can be represented by a vector of terms. Terms in this model are not equally weighted: each term is associated with a

Manuscript received September 10, 2007.
Yuan J. Lui is with the Computing Laboratory, University of Oxford, Oxford, OX1 3QD, UK (email: yuan.lui@st-hughs.ox.ac.uk).

specific weight which reflects the importance of that term. *Term frequency* (TF) and *inverse document frequency* (IDF) are the two most important term weights in this model [11].

TF is the frequency of a term in the document. The more often a term occurs in the document, the more likely it is to be important for that document. The *standard TF* of a term *T* in a document *D* is calculated by:

$$\text{Standard TF} = \text{no. of occurrences of } T \text{ in } D \quad (1)$$

IDF is the rarity of a term across the collection. A term that occurs in only a few documents is often more valuable than a term that occurs in many documents. The *standard IDF* of a term *T* is given by:

$$\text{Standard IDF} = \log \frac{\text{no. of documents in collection}}{\text{no. of documents } T \text{ occurs in}} \quad (2)$$

*TF×IDF* is a common way of combining TF and IDF. Despite the popularity of these weights, they do not have a universal definition.

Salton and Buckley [10] review the use of statistical information for weighting document terms and query terms, and discuss various ways of defining and combining TF and IDF. A total of 1,800 different term weighting combinations were used in their experiments, and 287 were found to be distinct. They make recommendations on the best combination in different situations. For technical documents (like the ones used in our experiments), they recommend using the *normalized TF* and the standard IDF. The normalized TF is calculated by normalizing the standard TF factor by the maximum TF in the vector (with the result in the range of 0.5 to 1.0):

$$\text{Normalized TF} = 0.5 + 0.5 \frac{\text{TF}}{\max \text{TF}} \quad (3)$$

### B. GenEx

Turney [16] proposes a keyphrase extraction algorithm called *GenEx* which consists of a set of parameterized heuristic rules that are fine-tuned by a genetic algorithm. During training, the genetic algorithm adjusts the rules' parameters to maximize the match between the output keyphrases and the target keyphrases. Table I shows the parameters used in GenEx.

GenEx has been trained on a set of journal articles and tested on a different set of journal articles, web pages and email messages. The experiments show that machine learning techniques can be used for keyphrase extraction and that GenEx generalizes well across collections. While GenEx is trained on a collection of journal articles, it *successfully* extracts keyphrases from web pages on different topics (by 'successfully', we mean the algorithm is capable of extracting at least one correct keyphrase from the document).

### C. Kea

Frank *et al.* [4] discuss another keyphrase extraction algorithm called *Kea* which is based on a naïve Bayes learning technique. The basic model of Kea involves two attributes: *TF×IDF* and *distance*.

The standard TF is used, but the IDF is defined differently. They calculate the IDF of a term *T* in a document *D* by (the counters start with one to avoid taking the logarithm of zero):

$$\text{Kea's IDF} = -\log(\text{no. of documents in collection that contain } T, \text{ excluding } D) \quad (4)$$

The *distance* attribute is the position where a term first appears in the document. A term that occurs at the beginning of the document is often more valuable than a term that occurs at the end of that document. The *distance* of a term *T* in a document *D* is given by:

$$\text{Distance} = \frac{\text{no. of words before first appearance of } T}{\text{no. of words in } D} \quad (5)$$

Kea uses the same set of training and testing documents as GenEx so that its performance can be directly compared with GenEx. The experiments indicate that GenEx and Kea perform at roughly the same level, measured by the average number of matches between author-assigned keyphrases and machine-extracted keyphrases [17].

### D. LAKE

D'Avanzo *et al*. [1], [2] propose a keyphrase extraction algorithm called *LAKE*. The algorithm uses two attributes, *TF×IDF* and *first occurrence* (same as *distance*), and some computational linguistics techniques to select candidate phrases.

LAKE selects candidate phrases in several steps: 1) Tag the input document. 2) Group sequences of words which are considered a single lexical unit together, e.g. 'Christmas' and 'tree' are combined into 'Christmas tree'. 3) Identify all the named entities in the document, e.g. 'London', 'IBM'. 4) Select candidate phrases from the document if they match one of the many manually predefined linguistics-based patterns, e.g. noun + verb + adjective + noun ('+' denotes 'followed by').

The experiments suggest that this algorithm works. Nevertheless, since LAKE uses a different set of training and testing documents, it is not certain if it is better than GenEx and Kea as it is not possible to directly compare their results. It is also because of this reason, LAKE has not been used as a standard of comparison for evaluating the performance of KE. All the keyphrase extraction tools in our experiments have been trained and tested on the same corpus so that direct comparison is possible.

### E. KPSpotter

Song *et al.* [12] discuss a keyphrase extraction system called *KPSpotter*. The system can process various formats of input data such as XML, HTML, and unstructured text data, and generate an XML file as output. It involves two attributes: *TF×IDF* and *Distance from First Occurrence* (same as *distance*). These numeric attributes are discretized into ranges and the resulting nominal attributes are used to calculate the information gain of each candidate phrase. The candidate phrases are then ranked in order of information gain.

KPSpotter has been trained and tested on a set of abstracts (rather than full documents) of technical reports. The same data have been used to train and test Kea so that the

performance of KPSpotter can be directly compared with Kea. The experiments show that KPSpotter and Kea give similar results. Nevertheless, since KPSpotter uses a different set of training and testing data (i.e. a collection of abstracts rather than documents), it is not possible to directly compare its results with ours. Therefore, KPSpotter has not been used as a standard of comparison in our experiments.

TABLE I
PARAMETERS USED IN GENEX

| Parameter | Description |
|---|---|
| NUM_PHRASES | Length of the output list, i.e. the number of keyphrases to be output |
| NUM_WORKING | Length of the working list, i.e. only words ranked higher than this are considered as candidate phrases |
| FACTOR_TWO_ONE | Reward for two-word phrases |
| FACTOR_THREE_ONE | Reward for three-word phrases |
| MIN_LENGTH_LOW_RANK | Low rank words must be longer than this; if not, they might be removed from the output list |
| MIN_RANK_LOW_LENGTH | Short words must be ranked higher than this; if not, they might be removed from the output list |
| FIRST_LOW_THRESH | Definition of early occurrence; words which first occur before this position are rewarded by FIRST_LOW_FACTOR |
| FIRST_HIGH_THRESH | Definition of late occurrence; words which first occur after this position are penalized by FIRST_HIGH_FACTOR |
| FIRST_LOW_FACTOR | Reward for early occurrence |
| FIRST_HIGH_FACTOR | Penalty for late occurrence |
| STEM_LENGTH | Maximum characters for fixed length stemming |
| SUPPRESS_PROPER | Flag for suppressing proper nouns |

### F. *Kea++*

Medelyan and Witten [8] propose a new method of improving the quality of the output keyphrases called *Kea++*. Kea++ is based on Kea, but differs from it in two ways: Kea++ uses a domain dependent thesaurus and a different set of attributes. Non-descriptors in the document are first replaced by their equivalent descriptors using semantic information about terms and phrases in the thesaurus. Descriptors and non-descriptors are synonyms. Descriptors refer to the 'preferred' terms, and non-descriptors refer to the 'less preferred' terms, e.g. 'love' is a descriptor and 'affection' is a non-descriptor. Candidate phrases are then measured by four attributes: *TF×IDF*, *distance*, *node degree*, and the length of a candidate phrase in words. The first two attributes are used in Kea. The *node degree* attribute is the number of thesaurus links that connect a candidate phrase to other candidate phrases.

Kea++ has been test on a set of documents on food and agriculture. The experiments indicate that Kea++ significantly outperforms Kea. Nevertheless, Kea++ has not been used as a standard of comparison in our experiments. Kea++ uses a controlled vocabulary list and is tied to a specific domain, whereas KE is a domain independent algorithm. In addition, Kea++ uses a different set of training and testing documents, so it is not possible to directly compare its results with ours.

### G. *W3SS*

Zhang *et al.* [19] introduce a new approach to automatic summarization of web sites called *W3SS*. The output summary is based on keywords and keyphrases extracted from the web site and is generated in several steps: 1) Get a set of web pages from a given site. 2) Remove all the tags and scripts in those pages and get a set of plain text. 3) Use the number of words in a paragraph and the part-of-speech of the words in a paragraph to extract narrative paragraphs from the plain text. 4) Use the part-of-speech of a word and the number of occurrences of a word in the narrative text, anchor text (e.g. hyperlinks) and special text (e.g. italic text) to extract keywords. 5) Use the keywords, the part-of-speech of a phrase and the number of occurrences of a phrase in the narrative text, anchor text and special text to extract keyphrases. 6) Use the extracted keywords and keyphrases to extract key sentences. 7) Provide the extracted keywords, keyphrases and key sentences as a summary of that site.

W3SS has been tested on a set of web sites. Human assessors are divided into a few groups and asked to answer questions about those sites (e.g. the purpose of a site). The experiments show that the group that read the manual summaries give the best results, followed by the group that read the generated summaries. Despite being interesting, W3SS has not been used as a standard of comparison in our experiments. All the documents used in our experiments are plain text, i.e. there is no anchor text and special text. The aim of W3SS is also different from ours: W3SS aims at summarizing a collection of web documents (i.e. web site), whereas KE aims at summarizing a single document.

## III. KEYPHRASE EXTRACTION

This section introduces the attributes used in the KE algorithm, gives an overview of KE, and compares KE with GenEx and Kea.

### A. *Attributes*

The selection of relevant attributes is probably the most important factor in determining the effectiveness of a

keyphrase extraction algorithm. Many attributes have been evaluated in our experiments, e.g. the length of a document, the number of characters in a term, the number of occurrences of a term in the collection, etc. However, only five of them have been found useful for keyphrase extraction:

- The *TF×IDF* attribute has already been discussed; see Section II-A for details.
- The *position* attribute is the same as Kea's *distance*; see Section II-C for details.
- The *title* attribute is a flag that indicates if a term appears in the title of the document. A term that occurs in the title of the document is often more valuable than a term that does not. Titles may not provide enough information on their own, but they may contain some important words. In fact, it has been reported that the use of abstracts in addition to titles brings substantial advantages in retrieval effectiveness and that the additional utilization of the full texts of the documents appears to produce little improvement over titles and abstracts alone in most subject areas [11]. If a term is found in the title, *title* is set to 1; otherwise, it is set to 0.
- The *proper noun* attribute is a flag that indicates if a term is a proper noun. If a term is a proper noun, *proper noun* is set to 1; otherwise, it is set to 0.
- The *number of terms* attribute is the number of terms in a term phrase.

### B. The KE algorithm

The KE algorithm is based on GenEx and Kea (for details of the differences between KE, GenEx, and Kea, see Section III-C) and consists of seven steps:

- Step 1 is to tag the input document and to select all the words which have been tagged as adjective, verb and noun and are not included in the stopword list. Although it is unlikely that adjectives and verbs will be output, they help to boost the score of their noun form (provided their stems are the same as the noun's) and therefore increase the likelihood that it will be output.
- Step 2 is to stem the selected words, to calculate the *TF×IDF*, *position*, *title* and *proper noun* of each term, to assign a score to each term based on these attributes, and to sort the terms in descending order of score (if two terms have the same score, they are ranked in ascending order of *position*).
- Step 3 is to select all the noun phrases in the document. Like KE, LAKE uses a part-of-speech tagger to select candidate phrases if they match one of the many manually predefined linguistics-based patterns (see Section II-D). Nevertheless, we believe this could be simplified by selecting only noun phrases, which can be naively defined as zero, one or two nouns or adjectives followed by a noun or a gerund, from the document. This is because almost all the keyphrases are noun phrases and they normally follow this definition [15].
- Step 4 is similar to Step 2. The main differences are that noun phrases, instead of words, are stemmed, the *TF×IDF*, *position*, *title*, and *number of terms* of each term phrase is calculated, and if two term phrases have the same score, they are ranked in ascending order of *position* followed by descending order of *number of terms*.
- Step 5 is to expand the single terms to term phrases. For each term, find all the term phrases that contain the term, and link it with the highest scoring term phrase. The result is a list of term phrases. The scores calculated in Step 2 are used to rank this list because it is generally preferable to represent documents and measure the importance of each representation element in terms of single terms rather than term phrases [10]. Term phrases, on the other hand, are used for output purposes. This is because documents are summarized by a set of phrases, not words.
- Step 6 is to eliminate duplicates from the list of term phrases. More than one term may be linked to the same term phrase. If that is the case, the term phrase will be linked to the highest scoring term.
- Step 7 is to identify the most frequent corresponding phrase in the document for each of the linked term phrases. If a term phrase is linked to more than one phrase, the most frequent phrase will be chosen. This step also eliminates subphrases if they do not perform better than their superphrases. If phrase $P_1$ occurs within phrase $P_2$, $P_1$ is a subphrase of $P_2$ and $P_2$ is a superphrase of $P_1$. If a phrase is a subphrase of another phrase, it will only be accepted as a keyphrase if it is ranked higher; otherwise it will be deleted from the output list.

### C. Comparison with GenEx and Kea

KE is based on GenEx and Kea, but differs from them in several ways:

- Purely statistical methods have been used in GenEx and Kea. KE, however, uses a combination of statistical and computational linguistics techniques for keyphrase extraction. Part-of-speech tagging, which is a useful computational linguistics technique, has been used to improve the quality of candidate phrases. Only words which have been tagged as adjective, verb and noun are selected as candidate phrases.
- KE uses a different set of attributes to discriminate between keyphrases and non-keyphrases: *TF×IDF*, *position*, *title*, *proper noun* and *number of terms*. Kea uses only two attributes: *TF×IDF* and *distance*. GenEx, on the other hand, uses many more attributes, but it does not use *TF×IDF* and *title*.
- KE uses a different machine learning algorithm; it is tuned by an artificial neural network (for details of the training of KE, see Section IV-B). GenEx is tuned by a genetic algorithm, whereas Kea is based on a naïve Bayes learning technique.

- KE is a different model; it consists of seven steps, and takes both words and phrases as candidate phrases. Kea is a simple model; it only selects phrases as candidate phrases, so it does not involve any linking between words and phrases. GenEx is more complicated; it consists of ten steps, considers both words and phrases, and involves many post-processing tasks.

The experimental results summarized in Table II suggest that these differences make KE a better algorithm than GenEx and Kea.

## IV. Experiments

This section explains how we evaluate the output keyphrases and train the KE algorithm, compares the individual performance of different attributes, the performance of different *TF×IDF* combinations and different keyphrase extraction tools, and the performance of KE on different learning methods and different corpora.

### A. Methodology

KE has been tested on two different corpora. The first corpus is the same as the one used in GenEx and Kea, and it has been used to train and test KE in all our experiments (except the one in Section IV-G). The criteria used for evaluating the output keyphrases are also the same as in GenEx and Kea (i.e. a machine-extracted keyphrase is said to be *correct* if its stem matches the stem of an author-assigned keyphrase), so direct comparison is possible. For details of this corpus and the evaluation method used, see [16]. The second corpus is different and larger than the first one, and it has been used to test the generalization performance of KE. The evaluation criteria are the same as the first corpus.

### B. Training of KE

The set of terms (i.e. output of Step 2) and the set of term phrases (i.e. output of Step 4) were tuned separately by a fully connected 4-9-1 back-propagation neural network. The resulting sets were then combined to perform Step 5, 6 and 7 of the KE algorithm. The number of hidden units affects the generalization performance of a neural network. We have tested different numbers of hidden units, and found that nine hidden units give the best result. Also, it is possible to have more than one hidden layer in a neural network, but one hidden layer is adequate for most applications. KE has been tuned and tested on a neural network with two hidden layers, but the difference between that and one hidden layer is not statistically significant. Therefore, only one hidden layer is used.

The experiments also indicate that the term set often requires more training iterations than the term phrase set. A training iteration involves all the documents in the training set and the selection of 150 terms (or term phrases), including both keyphrase and non-keyphrase examples, from each document. The cross-validation method has been used to estimate the appropriate point to stop training to avoid overfitting.

### C. Different attributes

Five different attributes are used in the KE algorithm, but we have only compared the individual performance of four attributes: *TF×IDF* (using the standard TF and Kea's IDF), *position*, *title*, and *proper noun*. *Number of terms* has not been evaluated in this experiment. Since *number of terms* is always one when it comes to single terms, the attribute (if used alone) cannot discriminate between different terms. Therefore, we decided not to evaluate the individual performance of this attribute.

Fig. 1 shows the comparison of the individual performance of different attributes with varying number of output keyphrases. *Precision* is the proportion of the keyphrases extracted that are correct. The experiments indicate that the performance of *position* is more stable than *TF×IDF*. The average precision of *position* lies between 0.21 and 0.25, whereas *TF×IDF* lies between 0.16 and 0.35. Also, there is a tendency for the average precision of *TF×IDF* to fall. The experiments also show that the performance of *position* is always better than *title*, and that *proper noun* gives the worst performance. We conclude that *position* is the best individual indicator of keyphrase extraction. This confirms the findings by Edmundson (1969) and Kupiec *et al.* (1995) that location-based methods give the best performance, though their work is concerned with sentence extraction and they use a different set of attributes. For details of their work, see [7].

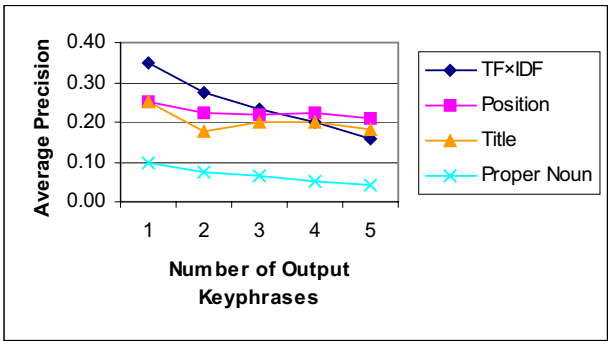


Fig. 1 Comparison of the individual performance of different attributes

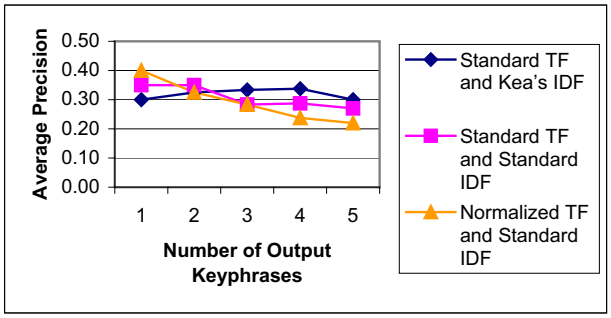


Fig. 2 Comparison of different combinations of TF×IDF

### D. Different TF×IDF combinations

As mentioned before, there is no universal definition of *TF×IDF*. Four different *TF×IDF* definitions have been discussed: standard TF, standard IDF, normalized TF, and

Kea's IDF. Three different combinations of *TF×IDF* have been implemented using these definitions and tested in our experiments.

Fig. 2 shows the comparison of different *TF×IDF* combinations with varying number of output keyphrases. The difference between the standard TF and Kea's IDF and the standard TF and standard IDF is not statistically significant, though the former tends to give more stable results. The average precision of the standard TF and Kea's IDF lies between 0.30 and 0.34, whereas the standard TF and standard IDF lies between 0.27 and 0.35. The average precision of the normalized TF and standard IDF lies between 0.22 and 0.40, and has a tendency to fall.

TABLE II
EXPERIMENTAL RESULTS FOR DIFFERENT KEYPHRASE EXTRACTION TOOLS

| | Average Number of Correct Keyphrases | Standard Deviation |
|---|---|---|
| KE | 1.50 | 1.32 |
| GenEx | 1.45 | 1.24 |
| C4.5 | 1.40 | 1.28 |
| Kea | 1.35 | 0.93 |
| Kea-C4.5 | 1.20 | 0.83 |
| Word 2000 | 0.85 | 0.93 |

TABLE III
EXAMPLES OF THE KEYPHRASES EXTRACTED BY KE

| | |
|---|---|
| Title | Brain Rhythms, Cell Assemblies and Cognition: Evidence from the Processing of Words and Pseudowords |
| Author-assigned Keyphrases | Brain theory, cell assembly, cognition, event related potentials, ERP, electroencephalograph, EEG, gamma band, Hebb, language, lexical processing, magnetoencephalography, MEG, psychophysiology, periodicity, power spectral analysis, synchrony |
| Machine-extracted Keyphrases (Top 5) | Words, processing, cell, cell assemblies, spatiotemporal activity patterns |
| | |
| Title | Precis of: The Roots of Thinking |
| Author-assigned Keyphrases | Analogical thinking, animate form, concepts, evolution, tactile-kinesthetic body |
| Machine-extracted Keyphrases (Top 5) | Thinking, **concept**, **tactile kinesthetic body**, hominid evolution, thesis |

### *E. Different keyphrase extraction tools*

We have compared the performance of KE with other keyphrase extraction tools: GenEx, C4.5[1], Kea, Kea-C4.5[2], and Microsoft Word 2000 (the *AutoSummarize*[3] feature). C4.5 and Kea-C4.5 have not been discussed because they have mainly been used as a standard of comparison for evaluating the performance of GenEx and Kea respectively. Please refer to [4], [16] for details of C4.5 and Kea-C4.5. Microsoft Word was chosen because it is a very popular word processing tool with the extraction of keywords and key sentences feature. Five keyphrases have been extracted from each testing document by these tools and compared with the corresponding author-assigned keyphrases. The number of output keyphrases is set to five because AutoSummarize always generates exactly five keyphrases. Also, unlike the other tools, AutoSummarize cannot be trained and the output keyphrases always contain exactly one word.

Table II shows the number of correct keyphrases identified by different keyphrase extraction tools. Results of GenEx, C4.5, Kea, and Kea-C4.5 are from [4]. The experiments indicate that KE (using the standard TF and Kea's IDF) performs better than the other tools (in terms of the average number of correct keyphrases) and that the difference between KE, GenEx, C4.5 and Kea is not statistically significant. Since Word 2000 can only extract five single words from each document and most of the keyphrases in the corpus contain more than one word, it is not surprising that Word 2000 gives the worst performance. Table III shows the keyphrases extracted by KE from two testing documents. Correct keyphrases are printed in bold.

### *F. Different learning methods*

In addition to neural networks, we have used the C4.5 decision tree learning algorithm [9] to tune KE. There are two reasons for doing this:

- Different machine learning methods should give approximately the same performance results, but some methods might be more suitable for keyphrase extraction than others. As shown in the experiment in Section IV-E, the choice of a learning method does affect the performance of a keyphrase extraction algorithm: GenEx and Kea give different results when they are tuned by different learning methods.
- The experiment in Section IV-E suggests that KE performs better than other keyphrase extraction tools. Nevertheless, the improvement in performance could be a result of the algorithm (and the selection of attributes) itself and/ or the learning method (i.e. neural networks) employed. Both GenEx and Kea have been tuned by the C4.5 learning method, and the tuned keyphrase extraction algorithms have been used as a standard of comparison for evaluating the performance

[1] C4.5 consists of a set of parameterized heuristic rules that are fine-tuned by the C4.5 decision tree learning algorithm. Some of these parameters are used in GenEx.

[2] Kea-C4.5 is a variation of Kea. The pre- and post-processing are the same as Kea. The only difference is that it uses the C4.5 decision tree learning technique, instead of a naïve Bayes learning technique.

[3] The AutoSummarize feature aims at extracting key sentences from a given document and is available from the *Tools* menu. The generation of keywords is actually a by-product of AutoSummarize. When AutoSummarize is used, it also fills in the *Keywords* field of the document's *Properties*, which is available from the *File* menu.

of GenEx and Kea. If KE is tuned by the C4.5 learning method, we can exclude the effect of neural networks and evaluate only the performance of the algorithm.

The C4.5 learning algorithm is an unstable classification algorithm, i.e. the constructed classifier (i.e. a decision tree) is sensitive to small changes to the training data, so bagging has been used to improve performance by reducing variance [14], [18]. Both GenEx and Kea have been tuned by 50 bagged C4.5 decision trees [4], [16]. To ensure comparability, the same has been carried out on KE: 50 bagged C4.5 decision trees were used to tune the term set and the term phrase set separately.

There are a number of options, which allow users of the C4.5 program [9] to improve decision tree performance, such as the *–c* option and the *–m* option. The *–c* option sets the confidence threshold for pruning, and the *–m* option sets the minimum number of examples needed to form a leaf of the decision tree.

We have evaluated the performance of different numbers of training examples and different values of *–c* and *–m*, and found that KE gives the best performance when 200 terms and 150 term phrases are selected from each training document with *–c* set to 50% and *–m* to 10. The experiments also indicate that, in general, simple trees give better results than bushy trees. We believe this is because bushy trees tend to be overtrained on the training set.

Fig. 3 shows the performance of KE and KE-C4.5 (i.e. KE tuned by the C4.5 learning method). The experiments indicate that the performance of KE is more stable than KE-C4.5 (the average precision of KE lies between 0.30 and 0.34, whereas KE-C4.5 lies between 0.27 and 0.35), and that KE often gives better performance results than KE-C4.5, except when the desired number of output keyphrases is set to one. We conclude that neural networks are better for keyphrase extraction than the C4.5 learning algorithm.

Although KE gives better results when it is tuned by neural networks, neural networks have been criticized for their poor interpretability (i.e. the level of understanding and insight provided by the model). It is difficult to extract classification rules from neural networks. The C4.5 learning algorithm, however, can do that easily. Although the performance of KE-C4.5 is not as good as KE, KE-C4.5 can help us to understand how a phrase has been classified as a keyphrase or a non-keyphrase in this experiment. Fig. 4 and Fig. 5 show the decision trees constructed from the term set and the term phrase set respectively. Decision nodes are represented by rounded rectangles. All the attributes of KE have been normalized, so they lie in the range of 0 to 1.

The process of term classification is simplified by seven rules (see Fig. 4). Terms, that first appear at the beginning of the document, appear in the title, and have been tagged as proper noun, are useful for identifying keyphrases. *TF×IDF*, however, is trickier; it depends on the values of other attributes, but, in general, large *TF×IDF* values are not preferred.

The process of term phrase classification is also simplified by seven rules (see Fig. 5). Term phrases, that first appear at the beginning of the document, appear in the title, or not in the title but contain more than one term, are useful for identifying keyphrases. Large *TF×IDF* values are again not preferred.

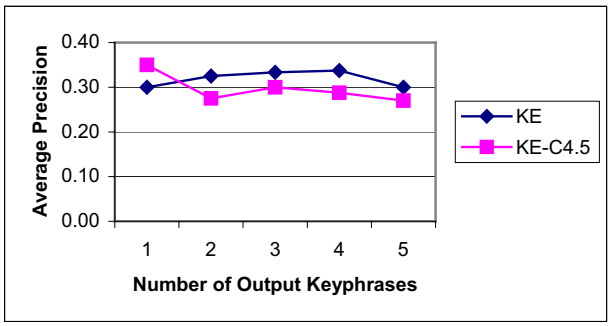


Fig. 3 Comparison of KE and KE-C4.5

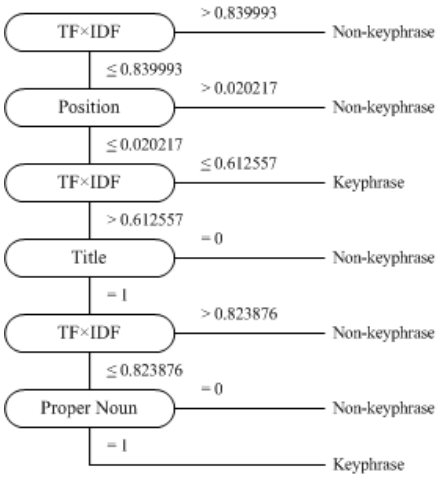


Fig. 4 Decision tree for classifying terms

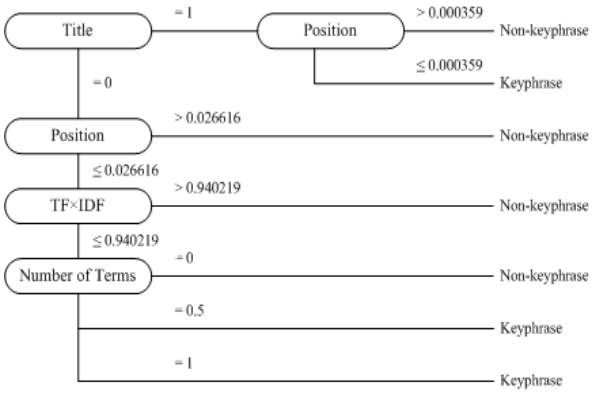


Fig. 5 Decision tree for classifying term phrases

### G. *Different corpus*

To ensure comparability, KE has been trained and tested on the same set of documents as GenEx and Kea. Nevertheless, we would like to see how KE performs when it is tested on a different, larger corpus. For convenience, we use *Corpus B* to refer to this corpus, and *Corpus A* to refer to the set of documents used by GenEx and Kea and in our previous

experiments.

Corpus B is used to evaluate the generalization performance of KE (tuned by neural networks using the training set in Corpus A). Corpus A and Corpus B are disjoint. Corpus B contains 231 articles selected from four journals. The journals are from different subject areas, including life sciences, mathematical sciences, and social sciences. Please see Table IV for the sources of Corpus B. All the articles contain keyphrases supplied by the authors.

To evaluate the correctness of the output keyphrases, we need a set of documents which contain author-assigned keyphrases. Journal articles are, as far as we know, the main source of these kinds of documents. It is not easy to find documents with author-assigned keyphrases in other areas. Even if some documents do contain keyphrases, the quality of these keyphrases might not be as good as those in journal articles. For example, keyphrases could be found in the *meta* tag of some web pages. However, these phrases are often unreliable and misleading, so most major search engines, including AltaVista, have stopped using them [13]. A recent study also confirms that the importance of these phrases to search engine ranking is little [3]. Therefore, journal articles have been used in this experiment.

Fig. 6 shows the comparison of the performance of KE on different journals with varying number of output keyphrases. KE does not seem to perform well in Corpus B compared with Corpus A. We believe this is because of the higher compression (or document-keyphrase) ratio in Corpus B. On average, there are 7936.83 words and 4.58 keyphrases per document in Corpus B compared with 4350.20 and 8.35 for the testing set in Corpus A. KE gives similar performance results on these journals, except when the desired number of output keyphrases is set to two. Most of the average precision of these journals lies around 0.20.

TABLE IV
SOURCES OF CORPUS B

| Journal Name | Field | Number of Documents |
|---|---|---|
| Journal of Molecular Biology | Molecular Biology | 46 |
| Information and Software Technology | Information Systems | 65 |
| Journal of Economic Behaviour and Organization | Economics and Econometrics | 57 |
| International Journal of Educational Development | Education | 63 |
| All | | 231 |

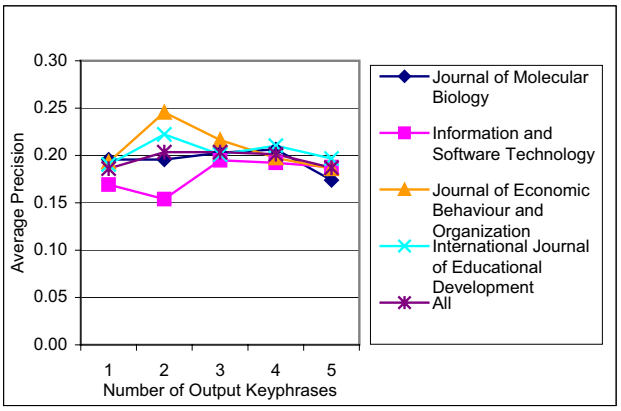


Fig. 6 Comparison of the performance of KE on different journals

## V. Discussion of Results

The above performance numbers are misleadingly low. Author-assigned keyphrases are often a small subset of the set of good quality keyphrases for a given document. On average, there are only 7.46 keyphrases per document in Corpus A (both training set and testing set) and 4.58 in Corpus B, and these phrases constitute less than 1% of the document length. A more accurate picture can be obtained by asking human assessors to evaluate the machine-extracted keyphrases. GenEx has been tested on 267 web pages: 62% of the keyphrases extracted from these pages are rated by human assessors as 'good', 18% as 'bad', and 20% as 'no opinion'. This suggests that about 80% of the keyphrases extracted by GenEx are acceptable [17]. The quality of machine-extracted keyphrases may not be as good as author-assigned keyphrases. Nevertheless, machine-extracted keyphrases could give the author a useful starting point for further manual refinement when author-assigned keyphrases are not available.

Some of the machine-extracted keyphrases are rather close to their corresponding author-assigned keyphrases, but because of the stemmer employed, they are regarded as different. For example, the author-assigned keyphrase 'cell assembly' is considered different from the machine-extracted keyphrase 'cell assemblies' (see the first example in Table III) because the stemmer maps 'assembly' to 'assemb' and 'assemblies' to 'assembl'. However, this kind of problem is inevitable if an automatic performance measure is used.

We notice that some common words are ranked fairly high in the output list despite the use of stopword lists and IDF. These words come from two main categories. Recall that the score of a term (or term phrase) is dependent on *TF*×*IDF*, *position*, and other attributes. Terms such as 'chapter' tend to occur at the beginning of the document. Early occurrence often boosts the score of these terms and increases the likelihood that they are output, though their IDF might be low. In addition, because of the nature of the corpora, terms such as 'person', which tend to occur rather frequently in everyday documents, appear only in a few documents. This boosts the IDF of these terms and improves their ranking. A possible way of solving this problem is to add these common words to

the stopword lists, but this will make KE more domain dependent, and that is not what we want.

The use of *proper noun* appears to degrade the performance of KE. This is probably because the training and testing documents are all academic papers, which tend to contain many proper nouns, especially in the References section. Indicator phrases [7] may be used to resolve this problem by ignoring all the words in the References section, but this will make KE more domain dependent. However, we believe that proper nouns might be useful in some domains (e.g. news) where they tend to occur less frequently, but further testing is needed to support this.

The domain independence of KE has also been confirmed in our experiments. KE successfully extracts keyphrases from documents on different subject areas (in Corpus B) while it has been trained on something totally different (i.e. training set in Corpus A).

Syntactic methods (e.g. the use of italics) seem helpful in extracting high quality keyphrases, and initially they were considered as an attribute for keyphrase extraction. However, all the documents in Corpus A are in ASCII and Unicode format, so we cannot implement this.

## VI. Conclusions and Future Work

We have discussed a new domain independent keyphrase extraction algorithm called KE, and shown that it performs better than other keyphrase extraction tools, including GenEx and Kea, and that it significantly outperforms Microsoft Word 2000's AutoSummarize feature. Machine-extracted keyphrases can provide valuable information about the content of a document, though they are not as good as author-assigned keyphrases. KE is currently targeted at the extraction of keyphrases from plain text, but it will be interesting to see if the use of hyperlink information in web documents can boost the quality of the output keyphrases.

## Acknowledgment

The author would like to thank his supervisors, Professor Richard Brent and Dr Ani Calinescu, for their valuable comments on his work.

## References

[1] E. D'Avanzo, B. Magnini and A. Vallin, "Keyphrase extraction for summarization purposes: the LAKE system at DUC-2004", Document Understanding Workshop, Boston, USA, 2004.
[2] E. D'Avanzo and B. Magnini, "A keyphrase-based approach to summarization: the LAKE system at DUC-2005", Document Understanding Workshop, Vancouver, Canada, 2005.
[3] R. Fishkin and J. Pollard, "Search engine ranking factors v2", http://www.seomoz.org/article/search-ranking-factors, 2007.
[4] E. Frank, G. Paynter, I. Witten, C. Gutwin and C. Nevill-Manning, "Domain-specific keyphrase extraction", Proceedings of 16th International Joint Conference on Artificial Intelligence, California, USA, Morgan Kaufmann, pp. 668-673, 1999.
[5] Y. Lui, "An improved keyphrase extraction algorithm", Proceedings of PREP2005, Lancaster, UK, 2005.
[6] Y. Lui, R. Brent and A. Calinescu, "Extracting significant phrases from text", Proceedings of IEEE Data Mining and Information Retrieval, Ontario, Canada, IEEE Computer, pp. 361-366, 2007.
[7] I. Mani, "Automatic summarization", John Benjamins, 2001.
[8] O. Medelyan and I. Witten, "Thesaurus based automatic keyphrase indexing", Proceedings of 6th ACM/ IEEE-CS Joint Conference on Digital Libraries, North Carolina, USA, ACM Press, pp. 296-297, 2006.
[9] R. Quinlan, "C4.5: programs for machine learning", Morgan Kaufmann, 1993.
[10] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval", Information Processing and Management, Vol. 24, No. 5, pp. 513-523, 1988.
[11] G. Salton and M. McGill, "Introduction to modern information retrieval", McGraw-Hill, 1983.
[12] M. Song, I. Song and X. Hu, "KPSpotter: a flexible information gain-based keyphrase extraction system", Proceedings of 5th ACM International Workshop on Web Information and Data Management, Louisiana, USA, ACM Press, pp. 50-53, 2003.
[13] D. Sullivan, "Death of a meta tag", http://searchenginewatch.com/showPage.html? page=2165061, 2002.
[14] P. Tan, M. Steinbach and V. Kumar, "Introduction to data mining", Addison-Wesley, 2006.
[15] P. Turney, "Extraction of keyphrases from text: evaluation of four algorithms", Technical Report ERB-1051, National Research Council of Canada, 1997.
[16] P. Turney, "Learning to extract keyphrases from text", Technical Report ERB-1057, National Research Council of Canada, 1999.
[17] P. Turney, "Coherent keyphrase extraction via web mining", Proceedings of 18th International Joint Conference on Artificial Intelligence, Acapulco, Mexico, CogPrints, pp. 434-439, 2003.
[18] I. Witten and E. Frank, "Data mining: practical machine learning tools and techniques with Java implementations", Morgan Kaufmann, 2000.
[19] Y. Zhang, N. Zincir-Heywood and E. Milios, "World wide web site summarization", Web Intelligence and Agent Systems, Vol. 2, Issue 1, pp. 39-53, 2004.