Extracting Significant Phrases from Text

Yuan J. Lui[*], Richard Brent[†] and Ani Calinescu[*]
[*]*University of Oxford, UK* [†]*Australian National University, Australia*

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 learning method to distinguish keyphrases from non-keyphrases. The experiments indicate that this algorithm performs at least as well as other keyphrase extraction tools and that it outperforms Microsoft Word 2000's AutoSummarize feature significantly.

1. Introduction

With the proliferation of the Internet and the huge numbers of documents ("document" is regarded as being synonymous with "text" in this paper) it contains, the provision of summaries of these documents has become more and more important. 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 [1-3, 8-10].

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 [10]. 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 [2].

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 of 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 at least as well as other keyphrase extraction tools and that it outperforms Microsoft Word 2000's AutoSummarize feature significantly.

Section 2 summarizes related work by other researchers. Section 3 introduces the KE algorithm and compares it with other keyphrase extraction

algorithms. The experimental results are presented in Section 4. Section 5 concludes this paper and discusses future work.

2. Related work

This section discusses two important term weights: term frequency and inverse document frequency, and two important keyphrase extraction algorithms: GenEx and Kea.

2.1. TF×IDF

The vector space model 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 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.

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:

Standard TF = no. of occurrences of *T* in *D*

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 calculated by:

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

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 (1988) 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 have been found to be distinct. They also 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 [6]:

$$\text{Normalized TF} = 0.5 + 0.5 \frac{\text{TF}}{\max \text{TF}}$$

2.2. GenEx

Turney (1999) 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. For details of the parameters used in GenEx, please refer to [9]. The experiments show that machine learning techniques can be used for the problem of 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.

2.3. Kea

Frank *et al.* (1999) discuss another keyphrase extraction algorithm called *Kea* which is based on the naïve Bayes learning technique [2]. The basic model 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:

Kea's IDF = -log (no. of documents in collection that contain *T*, excluding *D*)

The *distance* attribute is the position where a term first appears in the document. The *distance* of a term *T* in a document *D* is calculated by:

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

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 the author-assigned keyphrases and the machine-extracted keyphrases [10].

3. Keyphrase extraction

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

3.1. 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 considered, e.g. the frequency of a term, the length of a document, the position of a term in the document, etc. However, according to our experiments, only five attributes have been found useful for keyphrase extraction:

– The *TF×IDF* attribute has been discussed; please see Section 2.1 for details.

– The *position* attribute is the same as Kea's *distance*; please see Section 2.3 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 [7]. 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. A term phrase that occurs the same number of times as a term in the document is likely to be more valuable.

3.2. The KE algorithm

The KE algorithm is based on GenEx and Kea (for details of the differences between KE, GenEx, and Kea, please see Section 3.3) 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 shown.

– 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. The iterated Lovins algorithm has been used for stemming in our algorithm. It is chosen because it has been reported that aggressive stemming is better for keyphrase extraction than conservative stemming and that the iterated Lovins algorithm is more aggressive than the Porter and the Lovins algorithms [8, 9].

– Step 3 is to select all the noun phrases from the document. Like KE, D'Avanzo and Magnini (2005) use a part-of-speech tagger to help to select candidate phrases in their keyphrase extraction algorithm: candidate phrases are selected if they match one of the many manually predefined linguistics-based patterns, e.g. adjective + noun, and noun + verb + adjective + noun (the symbol '+' denotes 'followed by') [1]. 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 that they normally follow this definition [8].

– Step 4 is similar to Step 2. The main differences are that noun phrases, instead of words, are stemmed and that the *TF×IDF*, *position*, *title*, and *number of terms* of each term phrase is calculated.

– 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 [6]. 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 mapped by the highest scoring term.

– Step 7 is to identify the most frequent corresponding phrase in the document for each of the term phrase. If a term phrase is mapped by 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 the subphrase of P_2 and P_2 is the superphrase of P_1. If a phrase is the 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.

3.3. Comparison with GenEx and Kea

KE is based on GenEx and Kea, but is different 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 (i.e. 12 parameters), but it does not use *TF×IDF* and *title*.

– KE uses a different machine learning algorithm; it is trained by an artificial neural network (for details of the training of KE, please see Section 4.2). GenEx is trained by a genetic algorithm while Kea is based on the 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 mapping between words and phrases. GenEx is more complicated; it consists of ten steps, considers both words and phrases, and involves many post-processing tasks.

We expect these differences will make KE a better algorithm in some application domains, but further testing is needed to support this. Nevertheless, the results summarized in Table 1 (please see Section 4.5) suggest that KE is already at least as good as GenEx and Kea.

4. 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 combinations of *TF×IDF*, and the performance of different keyphrase extraction tools, and discusses the experimental results.

4.1. Methodology

KE has been trained and tested on the same set of documents as GenEx and Kea. Of course, this can be done on other corpora, but we will not able to compare our experimental results with other keyphrase extraction algorithms then. The criteria used for evaluating the output keyphrases are also the same as 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 the corpus and the evaluation method used, please refer to [9].

4.2. 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) have been trained separately by a fully connected 4-9-1 back-propagation neural network. The resulting sets are 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 [5]. 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 [5]. KE has been trained 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 [5] has been used to estimate the appropriate point to stop training to avoid overfitting.

4.3. 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 have decided not to evaluate the individual performance of this attribute.

Figure 1 shows the comparison of the individual performance of different attributes with varying number of output keyphrases. The experiments indicate that the performance of *position* is more stable than *TF×IDF* and 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 that they use a different set of attributes. For details of their work, please refer to [4].

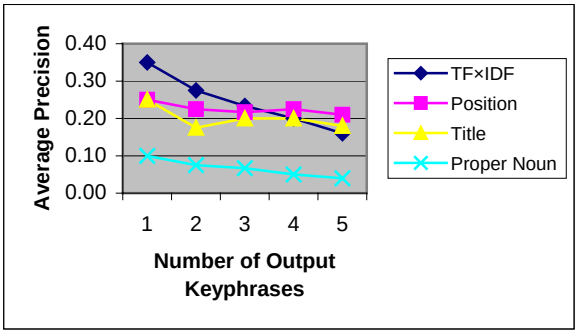

Figure 1. Comparison of different attributes' individual performance

4.4. Different combinations of TF×IDF

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.

Figure 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.

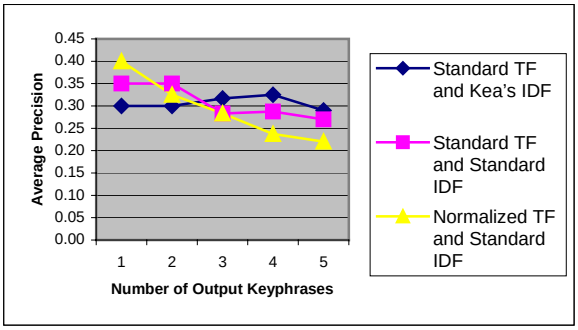

Figure 2. Comparison of different combinations of TF×IDF

4.5. Different keyphrase extraction tools

We have compared the performance of KE with other keyphrase extraction tools: GenEx, *C4.5*, Kea, *Kea-C4.5*, and Microsoft Word 2000 (the *AutoSummarize*[1] 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 [2, 9] for details of C4.5 and Kea-C4.5. 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 1 shows the number of correct keyphrases identified by different keyphrase extraction tools. Results of GenEx, C4.5, Kea, and Kea-C4.5 are from [2]. The experiments indicate that KE (using the standard TF and Kea's IDF) performs comparably to GenEx 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 that most of the keyphrases in the corpus contain more than one word, it is not surprising that Word 2000 gives the worst performance.

Table 1. Experimental results for different keyphrase extraction tools

	Average Number of Correct Keyphrases	Standard Deviation
KE	1.45	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

4.6. 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 7.5 keyphrases

[1] 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.

per document in the corpus, and these phrases constitute less than 0.001% 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 [9]. 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.

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 corpus, terms such as 'person', which tend to occur rather frequently in everyday documents, appear only in a few documents in the corpus. 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 [4] 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.

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

5. Conclusions and future work

We have discussed a new domain independent keyphrase extraction algorithm called KE, and shown that it performs at least as well as other keyphrase extraction tools, including GenEx and Kea, and that it outperforms Microsoft Word 2000's AutoSummarize feature significantly. Machine-extracted keyphrases could provide valuable information about the content of a document, though they are not as good as author-assigned keyphrases. To ensure comparability, KE has been trained and tested on the same set of documents as GenEx and Kea, but it will be interesting to see how KE performs when it is tested on a different (and possibly larger) corpus.

6. References

[1] E. D'Avanzo and B. Magnini, "A Keyphrase-Based Approach to Summarization: the LAKE System at DUC-2005", Document Understanding Workshop, Vancouver, Canada, 2005.

[2] 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, 1999, pp. 668-673.

[3] Y. Lui, "An Improved Keyphrase Extraction Algorithm", Proceedings of PREP2005, Lancaster, UK, 2005.

[4] I. Mani, "Automatic Summarization", John Benjamins, 2001.

[5] D. Rumelhart, B. Widrow and M. Lehr, "The Basic Ideas in Neural Networks", Communications of the ACM, Vol. 37, No. 3, 1994, pp. 87-92.

[6] G. Salton and C. Buckley, "Term-Weighting Approaches in Automatic Text Retrieval", Information Processing and Management, Vol. 24, No. 5, 1988, pp. 513-523.

[7] G. Salton and M. McGill, "Introduction to Modern Information Retrieval", McGraw-Hill, 1983.

[8] P. Turney, "Extraction of Keyphrases from Text: Evaluation of Four Algorithms", Technical Report ERB-1051, National Research Council of Canada, 1997.

[9] P. Turney, "Learning Algorithms for Keyphrase Extraction", Information Retrieval, Vol. 2, No. 4, 2000, pp. 303-336.

[10] P. Turney, "Coherent Keyphrase Extraction via Web Mining", Proceedings of 18th International Joint Conference on Artificial Intelligence, Acapulco, Mexico, CogPrints, 2003, pp. 434-439.