KPSpotter: A Flexible Information Gain-based Keyphrase Extraction System

Min Song
College of Information Science & Technology
Drexel University,
Philadelphia, PA 19104
(215) 895-2474, 01
E-mail: min.song@drexel.edu

Il-Yeol Song
College of Information Science & Technology
Drexel University,
Philadelphia, PA 19104
(215) 895-2489, 01
song@drexel.edu

Xiaohua Hu
College of Information Science & Technology
Drexel University,
Philadelphia, PA 19104
(215) 895-0551, 01
thu@cis.drexel.edu

ABSTRACT
To tackle the issue of information overload, we present an Information Gain-based KeyPhrase Extraction System, called KPSpotter. KPSpotter is a flexible web-enabled keyphrase extraction system, capable of processing various formats of input data, including web data, and generating the extraction model as well as the list of keyphrases in XML. In KPSpotter, the following two features were selected for training and extracting keyphrases: 1) TF*IDF and 2) Distance from First Occurrence. Input training and testing collections were processed in three stages: 1) Data Cleaning, 2) Data Tokenizing, and 3) Data Discretizing. To measure the system performance, the keyphrases extracted by KPSpotter are compared with the ones that the authors assigned. Our experiments show that the performance of KPSpotter was evaluated to be equivalent to KEA, a well-known keyphrase extraction system. KPSpotter, however, is differentiated from other extraction systems in the followings: First, KPSpotter employs a new keyphrase extraction technique that combines the Information Gain data mining measure and several Natural Language Processing techniques such as stemming and case-folding. Second, KPSpotter is able to process various types of input data such as XML, HTML, and unstructured text data and generate XML output. Third, the user can provide input data and execute KPSpotter through the Internet. Fourth, for efficiency and performance reason, KPSpotter stores candidate keyphrases and its related information such as frequency and stemmed form into an embedded database management system.

Categories and Subject Descriptors
D.2.8 [**Database Management**]: Database Applications – Data Mining

General Terms
Algorithms, Design, Experimentation

Keywords
Text Mining, Information Extraction, Summarization

1. INTRODUCTION
As more information is electronically available to us, we often encounter the problem of digesting unnecessary information in the course of seeking information we need. This information overload issue has been deteriorated by popularity of the Internet. A gigantic information depot, formed through the Internet, makes it difficult for us to consume the information.

In order to tackle the issue of "too much data but too little information", embedded in the large data collections, key phrases extraction or topic detection from a given data corpus is proposed by the text mining research community. Turney [5] defined the keyphrase extraction as the "automatic selection of important, topical phrases from the within the body of a document." As indicated in the definition, keyphrase extraction is different from "keyphrase assignment" in that it chooses keyphrases from the text, based on a model built with training data whereas keyphrase assignment uses the control vocabulary.

This paper describes KPSpotter that employs a new keyphrase extraction technique that combines the Information Gain data mining measure and several Natural Language Processing techniques such as stemming and case-folding. Information Gain is a well known data mining techniques introduced in ID3 algorithm [3]. KPSpotter is implemented in C++. With an object-oriented system architecture perspective, KPSpotter is developed as a flexible keyphrase extraction system to handle various types of file formats and to apply different stemming algorithms and data mining algorithms, whereas other keyphrase extraction systems such as KEA and GenEx require the input data to be certain formats. In addition, KPSpotter stores a training model as well as a list of keyphrases in XML tree that will be parsed and accessed through a web interface. For performance reason, KPSpotter stores candidate keyphrases and its related information such as frequency and stemmed form into an embedded database management system.

Our goal is to provide a flexible automatic keyphrase extraction system that generates useful metadata where none existed before. Although we evaluate the performance of KPSpotter by comparing the key phrases provided by the authors, we do not expect to equal them. If we can extract reasonable summaries from text documents, it will help the users understand the content of the collections.

The performance of KPSpotter was measured by comparing the number of matches of keyphrases that the authors assigned. We used the same training and test data that KEA used. We also conducted a simple performance comparison between KPSpotter and KEA. According to our preliminary experiments, the quality of keyphrases extracted by KPSpotter is equivalent to KEA's. From web-enabled technology perspective, KPSpotter is differentiated from KEA in that KPSpotter provides a web access for the user.

The remainder of this paper is as follows: Section 2 describes the system. Section 3 reports the evaluation results. Finally, section 4 concludes the paper.

2. SYSTEM DESCRIPTION

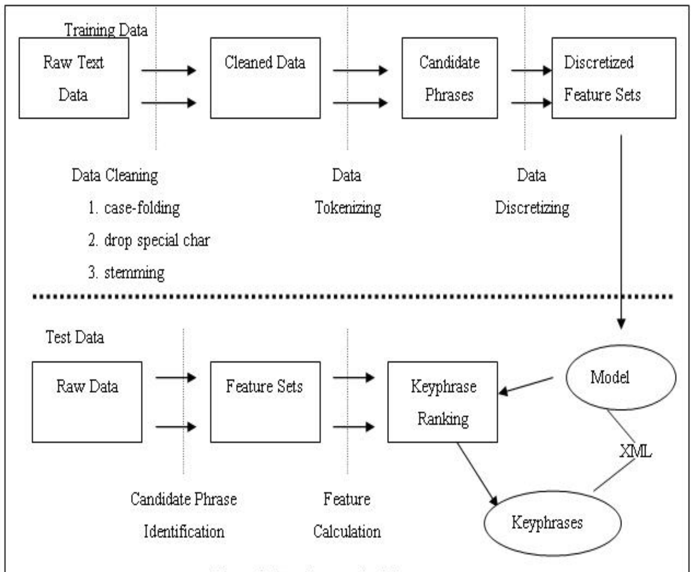

Figure 1. The System Architecture

In this section, we discuss how candidate keyphrases are extracted, an extraction model is built, and keyphrases for a new document are extracted. In addition, we explain the web interface of KPSpotter. KPSpotter comprises mainly two stages: 1) building extraction model and 2) extracting keyphrases. The figure 1 shows the overflow of the system. The detail descriptions are provided in the following subsections. These two stages are fully automated. Depending on the configuration parameters, KPSpotter processes either "building extraction model" mode or "extracting keyphrases" model. The outcomes of both processes by KPSpotter are stored in the XML form.

KPSpotter stores candidate keyphrases and its associated information in BerkeleyDB, an embedded database management system in the B-Tree data structure to gain better performance. This approach is particularly beneficial for the real application system to process a large number of candidate keyphrases. And KPSpotter is the first system that applies a persistent data storage technique to store and retrieve keyphrase candidates.

KPSpotter is evaluated with the training and test data provided by KEA. Both training and test data respectively consist of 80 abstracts from computer science technical reports available in the New Zealand Digital Library project.

2.1 Procedure of Selecting Candidate Phrases

In this section, we describe the selection criteria and rules for candidate phrases applied in KPSpotter.

Raw text data are filtered to regularize the text and determine initial phrase boundaries. The input stream is split into tokens (sequences of letters, digits, and internal periods). KPSpotter then considers all the subsequences in the document and determine which candidate phrases are suitable. Witten and his colleagues [6] came up with the following rules for candidate phrases:

1. Candidate phrases are limited to a certain maximum length (N consecutive words).
2. Candidate phrases cannot be proper names (i.e. single words that only ever appear with an initial capital)
3. Candidate phrases cannot begin or end with a stopword

KPSpotter adapts these rules for candidate phrases. Our stopword list consists of 150 unimportant words such as articles, prepositions, and conjunctions. All continuous sequences of words in each document are evaluated as candidate phrases with the three rules above. With regard to candidate selection criteria, KPSpotter is different from KEA in that KEA considers the contiguous sequences of words only in each input line, whereas KPSpotter extracts N consecutive words from the entire document.

2.2 Procedure of Feature Selection

In this section, we discuss the feature selection criteria used in KPSpotter. The following two feature sets were chosen and calculated for each candidate phrase: 1) TF*IDF and 2) Distance from First Occurrence.

*2.2.1 TF*IDF*

TF*IDF is proposed first by Salton and Buckley [4]. It is a measure of importance of term in a document or class. TF*IDF is term frequency in a document or class, relative to overall frequency. The TF*IDF feature was well-known weighting scheme in information retrieval and used in KEA.

2.2.2 Distance from First Occurrence

The distance from the first occurrence (FO) feature is calculated as the number of words that precedes the phrase's first appearance, divided by the number of words in the document.

2.2.3 Discretization

Since both features described above are continuous, we need to convert them into nominal to make our machine learning algorithm applicable. Among many discretization algorithms, we chose equal-depth (frequency) partitioning method which allows for good data scaling [1]. Equal-depth discretization divides the range into N intervals, each containing approximately the same number of samples.

During the training process, each feature is discretized. In KPSpotter, the value of each feature, a candidate phrase, is replaced by the range to which the value belongs. Table 1 shows the results of discretization by equal-depth partitioning method which is used for the training and testing data.

Feature	Discretization Ranges				
	1	2	3	4	5
TF*IDF	< 0.0031	>= 0.0031 && < 0.015	>= 0.015 && < 0.05	>= 0.05 && < 0.1	>= 0.1
Distance (FO)	< 0.15	>= 0.15 && < 0.35	>=0.35 && < 0.50	>= 0.50 && < 0.70	>= 0.70

Table 1: Discretization Table

During building a model, each phrase extracted from each training document is marked either keyphrases or non-keyphrases, by comparing with the actual keyphrases for the document provided by the author.

KPSpotter builds the model that predicts the class using the values of the two feature sets. The extraction model is stored in a XML file to make it portable. In the given sample model below, a XML element, called TUPLE, is a candidate phrase. Each tuple has two attributes, TF_IDF and DIS. These attributes have five values respectively. Each tuple also has a value indicating whether it matches with pre-determined keyphrases or not.

2.3 Extraction of Keyphrases

KPSpotter uses the Information Grain, GAIN(A), a measure of expected reduction in entropy based on the "usefulness" of an attribute A. It is one of the most popular measures of association currently used in data mining. Quinlan [3] use Information Gain for ID3 and its successor C4.5, a popular Decision Tree data mining technique. ID3 constructs simple trees by choosing at each step the splitting feature that "tells us the most" about the training data. Mathematically, the Information Gain is defined as Formula 3.

Each candidate phrase, extracted from a document, is ranked by probability calculated with GAIN(A). In KPSpotter, I(p,n) is stated such that class p: candidate phrase = "keyphrase" and class n: candidate phrase = "non-keyphrase."

For example, suppose that a candidate phrase, "text mining algorithm", gains 0.0134 for TF*IDF feature and it falls into the range 3 for TF*IDF. Let's also assume that it appears twice as a keyphrase in the class, tf_idf_3. The number of tuples in the tf_idf_3 is 20. The number of keyphrases is 120 out of the total 1000 candidate phrases. For this case, I(120,880) is 0.529361, E(tf_idf3) is 0.0469, and GAIN(Ptf_idf3) = 0.482462. It is ranked fifth as a keyphrase for the given test document.

KPSpotter generates a list of keyphrases for each document and stores it in a XML file (Table 2). As illustrated in Table 2, for each document identified by the filename (the value of the ID attribute is filename), a list of keyphrases extracted from the document with the weight calculated by Information Grain.

$$GAIN(A_i) = I(p,n) - E(A_i)$$

Where:

Ai: value of candidate phrase that falls into a discretized range

I(p,n): measures info required to classify any arbitrary tuple

$$\mathbf{I}(s_1, s_2, \ldots, s_m) = -\sum_{i=1}^{m} \frac{s_i}{s} \log_2 \frac{s_i}{s}$$

S contains si tuples of class Ci for i = {1, …, m}

E(W): Entropy of attribute W with values {a1,a2,…,av}

$$\mathrm{E}(W) = \sum_{j=1}^{v} \frac{s_{1j} + \ldots + s_{mj}}{s} I(s_{1j}, \ldots, s_{mj})$$

Formula 3. Information Gain

```
<KEY_PHRASE>
  <DOC ID="10050">
    <PHRASE RANK="0.566422">algorithm</PHRASE>
    <PHRASE RANK="0.590458">applic</PHRASE>
    <PHRASE RANK="0.544497">call</PHRASE>
    <PHRASE RANK="0.544497">causal</PHRASE>
    <PHRASE RANK="0.649069">causal multicast</PHRASE>
    <PHRASE RANK="0.544497">causal order</PHRASE>
    <PHRASE RANK="0.649069">cost</PHRASE>
    <PHRASE RANK="0.564413">messag</PHRASE>
    <PHRASE RANK="0.544497">multicast</PHRASE>
    <PHRASE RANK="0.566422">multicast algorithm</PHRASE>
    <PHRASE RANK="0.590458">multimedia</PHRASE>
    <PHRASE RANK="0.590458">multimedia applic</PHRASE>
    <PHRASE RANK="0.564413">order</PHRASE>
    <PHRASE RANK="0.544497">order multicast</PHRASE>
    <PHRASE RANK="0.566422">overhead</PHRASE>
    <PHRASE RANK="0.649069">present</PHRASE>
  </DOC>
</KEY_PHRASE>
```

Table 2: Sample Keyphrases Extracted by KPSpotter

3. EVALUATION

In this section, we report the preliminary experimental results of the performance of KPSpotter.

We measured the performance of KPSpotter by comparing key phrases with human-generated key phrases. Turney [5] reported that an average of about 75% of the human-generated keyphrases appears in the body of the corresponding document in his experiment data. With these findings, he argued that an ideal keyphrase extraction algorithm could generate phrases that match up to 75% of the author's keyphrases.

Taking this result into consideration, optimistically speaking, KPSpotter needs to extract three to four keyphrases matched from the list of keyphrases that the authors provided.

The overall performance of KPSpotter is illustrated further in Figure 2. For the given test documents, KPSpotter extracted one to two "correct keyphrases."

In Figure 2, the first line from the top is the average number of keyphrases that the authors assigned. The second line from the top shows the number of keyphrases that appears in the documents. The third one indicates the average number of correct identifications.

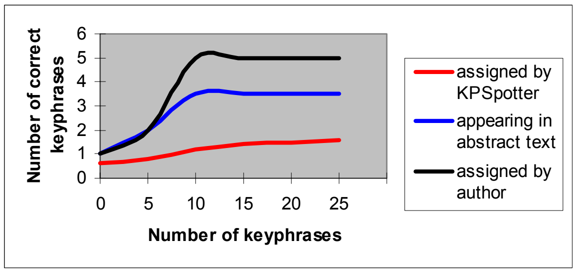

Figure 2: Overall Performance

A similar result was reported by KEA [2]. KEA generates about one to two "correct keyphrases" (Figure 3). Although KEA outperforms KPSpotter in the four ranges of keyphrase extraction, the margin is small, less than 0.2. In particular, the number of keyphrases to be extracted and delivered to the users by both systems will be the same.

The results from both the experiments, KPSpotter and KEA, seem to indicate that KPSpotter (and KEA) performs poorly in terms of average number of matches. There are several reasons influencing the performance. The primary influential factor is the document length of both the training and test sets. KEA reported that there was a big performance difference between the abstract-only document sets and full-text documents for the experiments.

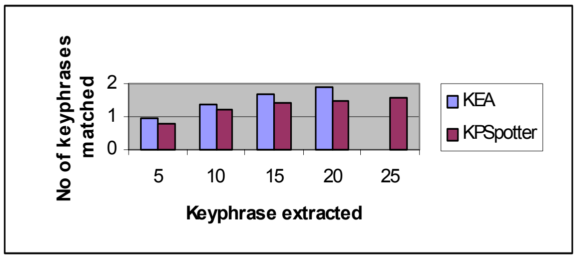

Figure 3: Performance comparison

We are undertaking in-depth experiments and analysis regarding influential factors on the performance.

4. CONCLUSION

We presented a flexible and automatic keyphrase extraction system, called KPSpotter, which employs a new technique combining the Information Gain data mining measure and several Natural Language Processing techniques such as stemming and case-folding. KPSpotter identifies candidate keyphrases using lexical methods, TF*IDF and distance from the first occurrence of the phrase.

In this paper, we also reported the preliminary experimental results. As discussed in the previous section, the quality of keyphrases extracted by KPSpotter is equivalent to KEA's in terms of accuracy, where both systems generate on average about one or two of the five keyphrases chosen by the author in the collections used for the experiment. Since the keyphrases chosen by the author do not necessarily represent the "key ideas" of the document, we argue that KPSpotter extracts a set of "semantically meaningful" keyphrases. More robust and sophisticated performance measures will be required to investigate our hypothesis. As mentioned in the paper earlier, we are currently undertaking this experiment.

KPSpotter is differentiated from other keyphrase extraction systems in that it introduces an extraction technique combining Information Gain and Natural Language Processing techniques, 2) it provides a web interface for the user to obtain a list of keyphrases for the supplied input data, 3) it processes various types of input data such as XML, HTML, and unstructured text data and generate XML output, 4) it stores statistical information of candidate phrases to BerkeleyDB, a persistent object storage device, and 5) it also stores both the model and list of keyphrases for the target document in a XML file.

These features of KPSpotter make it suitable for the real world application where robustness, flexibility, and speed are prerequisite.

We are also conducting an experiment of the performance comparison in predicting keyphrases. KPSpotter currently uses Information Gain and we are interested in how other information measures such as Mutual Information, Minimum Description Length predict the keyphrases from the list of candidate keyphrases.

5. REFERENCES

[1] Dougherty, J., Kohavi, R. and Sahami, M. (1995) Supervised and unsupervised discretization of continuous features. In: Proceeding of ICML-95, 12th International Conference on Machine Learning, Lake Tahoe, US, pp.194--202.

[2] Frank E., Paynter G.W., Witten I.H., Gutwin C. and Nevill-Manning C.G. (1999) Domain-specific keyphrase extraction, Proc. Sixteenth International Joint Conference on Artificial Intelligence, Morgan Kaufmann Publishers, San Francisco, CA, pp. 668-673.

[3] Quinlan, J. R. (1993) Programs for Machine Learning, San Mateo: Morgan Kaufmann Publishers, 1993.

[4] Salton G. and Buckley, C. (1988) Term-weighting approaches in automatic text retrieval, Information Processing and Management, pp.513-523.

[5] Turney, P.D. (2000) Learning algorithms for key phrase extraction. Information Retrieval, Information Retrieval, 2, 303-336.

[6] Witten I.H., Paynter G.W., Frank E., Gutwin C. and Nevill-Manning C.G. (1999) KEA: Practical automatic keyphrase extraction. In: Proc. DL '99, pp. 254-256.