Survey of Keyword Extraction Techniques

Brian Lott
bnlott@cs.unm.edu

December 4, 2012

Problem Description

Keywords are commonly used for search engines and document databases to locate information and determine if two pieces of test are related to each other. Reading and summarizing the contents of large entries of text into a small set of topics is difficult and time consuming for a human, so much so that it becomes nearly impossible to accomplish with limited manpower as the size of the information grows. As a result, automated systems are being more commonly used to do this task.

This problem is challenging due to the intricate complexities of natural language, as well as the inherent difficulty in determining if a word or set of words accurately represent topics present within the text.

With the advent of the internet, there is now both a massive amount of information available, as well as a demand to be able to search through all of this information. Keyword extraction from text data is a common tool used by search engines and indexes alike to quickly categorize and locate specific data based on explicitly or implicitly supplied keywords.

Motivation

Many different methods have been used over the years, and new solutions are constantly being proposed to solve this complex problem. A broad overview of the common techniques and algorithms has not yet been explored.

Methods

Various methods of locating and defining keywords have been used, both individually and in concert. Despite their differences, most methods have the same purpose and attempt to do the same thing: using some heuristic (such as distance between words, frequency of word use, or predetermined word relationships), locate and define a set of words that accurately convey themes or describe information contained in the text.

Word Frequency Analysis

Much early work concerned the frequency of term usage in the text, but most of this work focused on defining keywords in relation to a single document. In 1972, the idea of statistically analyzing the frequency of keyword usage within a document in relation to multiple other documents became more common. [4]

This technique, known as Term Frequency - Inverse Document Frequency or simply TF-IDF, weights a given term to determine how well the term describes an *individual* document within a corpus. It does this by weighting the term positively for the number of times the term occurs within the specific document, while also weighting the term negatively relative to the number of documents which contain the term. Consider term t and document $d \in D$, where t appears in n of N documents in D. The TF-IDF function is of the form:

$$TFIDF(t, d, n, N) = TF(t, d) \times IDF(n, N) \tag{1}$$

There are many possible TF and IDF functions. Practically, nearly any function could be used for the TF and IDF. Regularly-used functions include [9]:

$$TF(t, d) = \begin{cases} 1 & \text{if } t \in d \\ 0 & \text{else} \end{cases} \tag{2}$$

$$TF(t, d) = \sum_{word \in d} \begin{matrix} 1 & \text{if } word = t \\ 0 & \text{else} \end{matrix} \tag{3}$$

Additionally, the term frequency may be normalized to some range. This is then combined with the IDF function. Examples of possible IDF functions include:

$$IDF(n, N) = \log\left(\frac{N}{n}\right) \tag{4}$$

$$IDF(n, N) = \log\left(\frac{N - n}{n}\right) \tag{5}$$

Thus, a possible resulting TFIDF function could be:

$$TFIDF(t, d, n, N) = \left(\sum_{word \in d} \begin{array}{ll} 1 & \text{if } word = t \\ 0 & \text{else} \end{array} \right) \times \log \left(\frac{N - n}{n} \right) \quad (6)$$

When the TF-IDF function is run against all terms in all documents in the document corpus, the words can be ranked by their scores. A higher TF-IDF score indicates that a word is both important to the document, as well as relatively uncommon across the document corpus. This is often interpreted to mean that the word is significant to the document, and could be used to accurately summarize the document [4].

TF-IDF provides a good heuristic for determining likely candidate keywords, and it (as well as various modifications of it) have been shown to be effective after several decades of research. Several different methods of keyword extraction have been developed since TF-IDF was first published in 1972, and many of these newer methods still rely on some of the same theoretic backing as TF-IDF. Due to its effectiveness and simplicity, it remains in common use today [8].

Word Co-Occurrence Relationships

While many methods of keyword extraction rely on word frequency (either within the document, within the corpus, or some combination of these), various possible problems have been pointed out with these metrics [12] [5], including reliance on a corpus, and the assumption that a good keyword will appear frequently within the document but not within other documents within the corpus. These methods also do not attempt to observe any sort of relationship between words in a document.

Using a Document Corpus

One attempt at using this extra information utilizes a Markov Chain which is used to evaluate every word in the corpus of all documents [12]. This technique defines a Markov Chain for document d and term t with two

states (C, T) where the probability of transitioning from C to T is the probability that the given term was observed in documents d out of all documents (effectively the number of times that t occurs in d divided the number of times t occurs in all documents), while the probability of moving from T to C is the probability that the term was observed out of all terms in d (the number of times t occurs in d divided by the number of term occurrences in d. Conceptually, if two terms arrive at the same state with similar regularity, they are related.

The authors of this technique determined that a word is less likely to be descriptive of the document if it arrives at the same state with a similar frequency to many other words in the document (called the *background distribution*), while it is more likely to be descriptive of the document if it diverges the most from the background distribution. This technique was shown to match, and regularly beat, TF-IDF in terms of precision when run over a corpus of document abstracts from ACM [12].

Frequency-Based Single Document Keyword Extraction

Most methods of keyword extraction rely on using some method of comparing a document to a corpus to determine which words are most unique to an individual document. This measure becomes more difficult to use when the corpus is small, non-existent, or of a similar subject and composition.

One method developed by Matsuo and Ishizuka [5] to extract keywords from a single document uses word co-occurrence to build a co-occurrence matrix such as the one in Table 1. When using this method, two words are said to co-occur if they are both observed in a section of text delimited by a punctuation mark (effectively a sentence). In the given example, we can see that words b and c occur in the same sentence a total of 42 times in the document.

	a	b	c	d
a		5	13	7
b	5		42	3
c	13	42		25
d	7	3	25	

Table 1: Example co-occurrence matrix

The authors postulate that words are important to the document if they co-occur with other words more often in the document than they would if every instance of the word were randomly distributed. For some word w_i, this can be thought of as the ratio of the number of co-occurrences of words w_i, w_j to the number of all other co-occurrences involving w_i. Under the given assumptions, a high ratio would mean that the word w_i is a likely keyword for the document.

One clear problem would be if a word only occurs once in the document. The ratio value would not be based on enough information to be statistically significant. This ratio could also be unexpectedly high, since its row in the co-occurrence array would be extremely entirely sparse. To combat this, the authors use a Pearson's chi-squared test (also known as an X^2 value) for each word in the document.

$$\begin{aligned} \text{let } n &= \text{number of words} && (7)\\ \text{let } O &= \text{observed frequency} && (8)\\ \text{let } E &= \text{expected frequency} && (9)\\ X^2 &= \sum_{i=0}^{n} \left(\frac{(O_i - F_i)^2}{E_i} \right) && (10) \end{aligned}$$

This test allows the frequency distribution of each word to be tested and compared to an expected distribution. The authors expected a random distribution of words, and compared the observed distribution to the expected.

A word which occurs a small number of times would have an occurrence distribution close to the expected (random) distribution and would have a low X^2 value, while a word that occurs frequently and regularly co-occurs with another word would have a high x^2 value.

The authors showed that this technique was able to closely match TF-IDF, but did not rely on the use of a document corpus.

Content-Sensitive Single Document Keyword Extraction

Another method of keyword extraction was developed by Ohsawa, et al. [7], attacks this problem from a different angle. While many methods of keyword

extraction rely on statistical information gathered from term occurrence frequency in the document, this method, called KeyGraph, relies instead on clustering of related items into groups to determine which words in a document are representative of the document's content.

KeyGraph builds a graph representation of the document, with terms as nodes, and edges between nodes representing commonly occurring co-occurrences occurring within the document. *Clusters* of words are then identified by locating maximally-connected subgraphs within the document graph. Candidate keywords are then identified by locating nodes within the graph that have edges between two separate clusters. Intuitively, these candidate keywords are terms that join separate ideas or concepts (clusters), which the author[s] of the document in question wrote when he had both concepts in mind. These candidate keywords are then ranked by the probability that for each of the clusters they join, that word was the word used to join the two clusters (effectively, the most common word used used to join these clusters).

Tests of KeyGraph shows that it was able to match, and surpass, TF-IDF in a series of tests run by its creators [7]. Additionally, a series of tests run on social media data collected during the 2008 presidential election showed that KeyGraph was able to locate keywords in a noisy environment with large amounts of irrelevant information [10].

Keyword Extraction Using Lexical Chains

Lexical chains are simply a list of related words found in a text. The relationships are usually semantic, such as synonymy, hyponymy, or meronymy. One example of a lexical chain would be the following [6]:

$$\text{sugar maple} \rightarrow \text{maple} \rightarrow \text{tree} \rightarrow \text{plant} \tag{11}$$

This representation of semantic information in natural languages allows for context to be encoded in the structure of the chain. In the given example, this can be seen by the word "plant" following the word "tree". The word "plant" follows "tree" since "plant" is related to "maple". If not for this semantic information, the next word in the chain might have been something concerning graphs or data structures.

Lexical chains have regularly found use in automated text summarization techniques [1] [11] where they can be used to quickly and accurately locate terms and sequences of terms with similar meanings. Miller [6] gives the example: "A sugar maple is a maple that...". By following the example lexical chain, we can clearly see that "sugar maple" and "maple" have very similar meanings, and one of them may be a candidate for removal from the text for the purpose of creating a more concise summary of the phrase.

It was proposed by Ercan and Cicekli that lexical chains could be used to locate words important to a text [2]. Their technique relies around using a statistical classifier (C4.5) to build decision trees which can be used to determine whether a given word is a likely keyword. To do this, they assign each term in the text to the lexical chains which contain that term. Terms are then assigned scores based on the first and last locations within the document where the term is used, the average frequency at which the term appears, the first and last locations within the document where synonyms, hypernyms/hyponyms, and meronyms occur.

By using C4.5 with these scores, the authors were able to locate keywords within the text that matched author-supplied keywords with up to 64% accuracy.

Keyphrase Extraction Using Bayes Classifier

An adaptation of TF-IDF was used in conjunction with a naive bayes classifier by Frank et al. [3] to locate keyphrases in a document within a corpus. This method works by running the TF-IDF variation in equation 12 over every phrase in the document.

$$TFIDF(p, d) = \Pr\left[\text{phrase in } d \text{ is } p\right] \times -\log \Pr\left[p \text{ appears in any document}\right] \tag{12}$$

where p is the phrase in question, and d is the current document. The probability that each phrase in the document is a keyphrase is then determined using Bayes' theorem:

$$\Pr[key \mid T, D] = \frac{\Pr[T \mid key] \times \Pr[D \mid key] \times \Pr[key]}{\Pr[T, D]} \tag{13}$$

where T is the TF-IDF value computed earlier, and D is the distance into the document of the first occurrence of the given phrase (the number of phrases that appear before it). Thus, $\Pr[T \mid key]$ is the probability that the phrase in question has the TF-IDF value T, $\Pr[D \mid key]$ is the probability that the phrase occurs at distance D into the document in question, and $\Pr[key]$ is the probability the phrase is a keyphrase, out of all phrases in the document. $\Pr[T, D]$ is used to normalize the resulting value to fall in the range $[0, 1]$.

The phrases are then ranked by the probabilities that they are keyphrases given T, D, and the k desired keyphrases are extracted from the top k phases in the ranking.

The authors showed that this method performed either comparably or slightly better than contemporary methods in a series of tests against a collection of websites and a set of medical journal articles [3].

Conclusion

TF-IDF is one of the best-known and most commonly used keyword extraction algorithms currently in use [8] when a document corpus is available. Several newer methods adapt TF-IDF for use as part of their process, and many others rely on the same fundamental concept as TF-IDF. Nearly all keyword extraction algorithms which make use of a document corpus depend on a weighted function which balances some measure of term or phrase appearance within a document (frequency, location within document, co-occurrence with other words) with some similar measure from the corpus.

When a corpus is unavailable, keyword extraction techniques must usually make use of additional measurements in addition to those used by TF-IDF and related methods. Additional information sources include some form of lexical or semantic analysis, or some co-occurrence measure.

Bibliography

[1] R. Barzilay, M. Elhadad, et al. Using lexical chains for text summarization. In *Proceedings of the ACL workshop on intelligent scalable text summarization*, volume 17, pages 10–17, 1997.

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

[3] Eibe Frank, Gordon W. Paynter, Ian H. Witten, Carl Gutwin, and et al. Domain-specific keyphrase extraction. In *PROC. SIXTEENTH INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE*, pages 668–673. Morgan Kaufmann Publishers, 1999.

[4] K.S. Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972.

[5] Y. Matsuo and M. Ishizuka. Keyword extraction from a single document using word co-occurrence statistical information. *International Journal on Artificial Intelligence Tools*, 13:2004, 2004.

[6] G.A. Miller et al. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[7] Yukio Ohsawa, Nels E. Benson, and Masahiko Yachida. Keygraph: Automatic indexing by co-occurrence graph based on building construction metaphor. In *Proceedings of the Advances in Digital Libraries Conference*, ADL '98, pages 12–, Washington, DC, USA, 1998. IEEE Computer Society.

[8] S. Robertson. Understanding inverse document frequency: on theoretical arguments for idf. *Journal of Documentation*, 60(5):503–520, 2004.

[9] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[10] H. Sayyadi, M. Hurst, and A. Maykov. Event detection and tracking in social streams. In *Proceedings of International Conference on Weblogs and Social Media (ICWSM)*, 2009.

[11] H.G. Silber and K.F. McCoy. Efficiently computed lexical chains as an intermediate representation for automatic text summarization. *Computational Linguistics*, 28(4):487–496, 2002.

[12] Christian Wartena, Rogier Brussee, and Wout Slakhorst. Keyword extraction using word co-occurrence. In *Proceedings of the 2010 Workshops on Database and Expert Systems Applications*, DEXA '10, pages 54–58, Washington, DC, USA, 2010. IEEE Computer Society.