Aristotle University of Thessaloniki
Department of Informatics

Natural Language Processing Project 2024-03-25

Automatic Keyword Extraction from Scientific Documents

by

George Matlis
AEM 178

Abstract:

This report focuses on the automatic keyword extraction from scientific documents using the Rapid Automatic Keyword Extraction (RAKE) [2] method. RAKE, a robust and unsupervised method, is explored as the primary tool for identifying keywords in scientific abstracts. The report provides insights into RAKE's effectiveness, its simplicity, and its applicability to computer science document abstracts.

Keywords: RAKE, Keyword Extraction

1 Introduction

Keywords, which are defined as a string of one or more words, serve as a concise representation of a document's content. Widely employed within information retrieval (IR) systems, keywords prove advantageous due to their ease of definition, revision, recollection, and sharing. Despite their usefulness for analysis, indexing, and retrieval, a majority of documents lack assigned keywords. Existing approaches often center around manual keyword assignment by professionals, who may adhere to a fixed taxonomy or rely on the author's judgment to compile a representative list. Consequently, research has concentrated on methods to automatically extract keywords from documents, serving either as a guide for professional indexers or generating summary features for otherwise inaccessible documents.
The subsequent sections of this report describe the Rapid Automatic Keyword Extraction (RAKE), an unsupervised and language-independent approach for extracting keywords from abstracts of scientific documents. Detailed insights into the algorithm and its configuration parameters are presented, accompanied by results obtained from a benchmark dataset of computer science abstracts. A comparative analysis is conducted, referencing the results from [2]. Additionally, an effort is made to reproduce the stoplists generated in [2], employed for configuring RAKE.

2 RAKE

RAKE is based on the observation that keywords frequently contain multiple words but rarely contain standard punctuation or stop words. Stop words are function words like *and*, *the*, *of*, or any other word with little to no meaning. These words are typically excluded in keyword extraction applications as they lack meaningful contributions to the document's content.
RAKE uses the stoplist (the words labelled as stop words) as input, along with a set of phrase delimiters, and a set of word delimiters. RAKE initially extracts the keywords from a document by parsing its text into a set of candidate keywords using the stoplist, and the set of word and phrase delimiters. More specifically, the document text is split into an array of words using the word delimiters. This array is then split into sequences of contiguous words at phrase delimiters and stop word positions. Words within a sequence are assigned the same position in the text and together are considered a candidate keyword. Figure 1 depicts the entire procedure of RAKE.

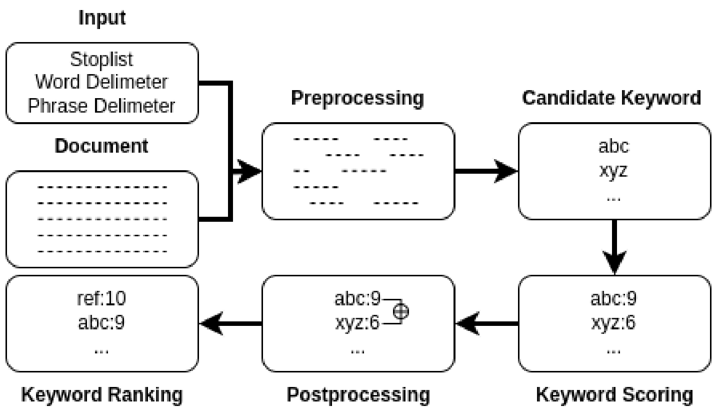

Figure 1: RAKE extracts candidate keywords, scores them based on individual word scores, and ranks them. Post-processing, though optional, may include semantic enhancements to further optimize the scoring of candidate keywords in RAKE.

As an example, consider the following abstract from the corpus:

Compatibility of systems of linear constraints over the set of natural numbers.

Criteria of compatibility of a system of linear Diophantine equations, strict inequations, and nonstrict inequations are considered. Upper bounds for components of a minimal set of solutions and algorithms of construction of minimal generating sets of solutions for all types of systems are given. These criteria and the corresponding algorithms for constructing a minimal supporting set of solutions can be used in solving all the considered types of systems and systems of mixed types.

The manually assigned keywords are:

linear constraints, set of natural numbers, linear Diophantine equations, strict inequations, nonstrict inequations, upper bounds, minimal generating sets

The candidate keywords extracted by RAKE along with their scores are listed in Table 1. The authors in [2] do not explicitly specify whether they utilize both the title and the abstract of the document for extracting keywords. In my implementation, i incorporate both, deeming the title highly valuable for extracting candidate keywords due to its often content-bearing nature, implying that the words comprising the title are meaningful and useful for defining keywords.

Score	**Keyword**	**Score**	**Keyword**
8.7	minimal generating sets	8.5	linear diophantine equations
7.7	minimal supporting set	4.7	minimal set
4.5	linear constraints	4.0	upper bounds
4.0	strict inequations	4.0	nonstrict inequations
4.0	natural numbers	3.7	mixed types
3.5	corresponding algorithms	3.2	considered types
2.0	set	1.7	types
1.5	considered	1.5	algorithms
1.0	used	1.0	systems
1.0	system	1.0	solving
1.0	solutions	1.0	given
1.0	construction	1.0	constructing
1.0	components	1.0	compatibility

Table 1: Candidate keywords with their scores

The score for each candidate keyword is computed as the sum of its member word scores. The word scores are calculated using the degree to frequency ratio obtained and computed from the word co-occurrence graph. The co-occurrence graph encompasses each unique word extracted by RAKE, computing both the word degree and word frequency metrics. Word degree considers all instances of a word, including its co-occurrence with other words in the document. On the other hand, word frequency simply enumerates the number of times a word appears in the document. The degree to frequency ratio favors words that occur in long candidate keywords. The word scores are depicted in Table 2.

Upon analyzing Table 2, it becomes apparent that certain words exhibit high similarity, such as *set* and *sets*. In some instances, two words may serve as synonyms. To augment the semantic relevance captured by RAKE, i propose a modified computation of word degree and frequency metrics to aggregate the occurrences of similar words. Specifically, i modify RAKE to:

1. Identify synonyms for each word in the dictionary of word degrees and word frequencies

Word	deg(w)/freq(w)	Word	deg(w)/freq(w)
compatibility	1	systems	1
linear	2.5	constraints	2
set	2	natural	2
numbers	2	criteria	1
system	1	diophantine	3
equations	3	strict	2
inequations	2	nonstrict	2
considered	1.5	upper	2
bounds	2	components	1
minimal	2.7	solutions	1
algorithms	1.5	construction	1
generating	3	sets	3
types	1.7	given	1
corresponding	2	constructing	1
supporting	3	used	1
solving	1	mixed	2

Table 2: Word scores

2. Find the plural form of those synonyms

3. Verify the presence of synonyms in the dictionary of word degrees and word frequencies, then aggregate their respective scores

This approach aims to emphasize the significance of similar words in the RAKE scoring of candidate keywords. While this idea was implemented in the project, the outcomes were not substantial enough to be incorporated into this report. An alternative approach to infusing semantic meaning into RAKE entails assessing the similarity among the final scored keywords. This can be achieved by representing multi-word keywords as embeddings and comparing them. However, this method was not implemented in this project due to its complexity and is only mentioned for reference.

3 Stoplist Generation

A stoplist, or a stopword list, is a predefined list of words that are considered common, frequent, or irrelevant in a given context and are typically excluded from text processing or analysis. These words are often removed during tasks like text mining, information retrieval, or natural language processing to focus on more meaningful content. Stoplists usually include words such as articles, prepositions, and conjunctions that don't carry significant semantic meaning and are thus deemed less relevant for certain applications. In practice, stoplists can be hand-tuned for specific applications. domains. or languages. In the context of specific applications, such as the automatic extraction of keywords from scientific documents, a custom stoplist may vary from one designed for a similar purpose involving different document types. Consequently, creating a tailored stoplist for a particular application is often more advantageous than relying on a generic stoplist.
To generate stoplists, I partitioned the overall scientific document collection into a training set and a testing set. The training set comprised 1000 documents, while the testing set comprised 500 documents. RAKE was assessed using the testing set. Four metrics were employed in the generation of stoplists:

1. **Term Frequency:** The number of times a word appears in all documents in the training set.

2. **Document Frequency:** The number of documents that contain a word. This metric can be computed by first creating the vocabulary V of the training set (all the possible words in the training set), and for each word in V, count the number of documents that contain that word.

3. **Keyword Frequency:** The number of times a word occurred within an abstract's manually assigned keywords. For example, if w is the word *the*, count the number of times the word *w* occurred withing the context of all the manually assigned keywords in the training set.

4. **Adjacent Frequency:** The number of times the word occurred adjacent to an abstract's manually assigned keywords. This metric is somewhat ambiguous because there are many ways of interpreting which words are adjacent to the manually assigned keyword. Let's consider a simple example. Given a manual keyword k consisting of more than one words, and a word w, computing the adjacent frequency of w can be done in the following ways:

- Count the number of times the pattern $w+"\ \ "+k$ occurs in the abstract that the keyword refers to. This pattern may be considered a single match (if it occurs once or multiple times) or multiple matches (n times if the pattern repeats n times in the abstract). Variations of the pattern, such as $w+"\ \ "+k+"\ \ "+w$ or $k+"\ \ "+w$, can be adjusted accordingly, with counting performed as outlined.

- Considering that k consists of more than one words, we can split k and apply the previous method of pattern matching for each word. The application of this method depends on whether the words of k are meaningful and significant.

Creating a stoplist solely based on term frequency may inadvertently include meaningful words, potentially leading to their exclusion from the analysis. Therefore, a reasonable hypothesis is that words adjacent to manually assigned keywords are less likely to convey significant meaning. As a result, they might be suitable candidates for inclusion in a stoplist. The table in 3 lists the top 10 stop words extracted using the four metrics. In Table 3, it is evident that stopwords are less likely to occur within a manually assigned keyword (Keyword Frequency) than adjacent to it (Adjacent Frequency). This underscores the importance of the Adjacent Frequency metric in generating a stoplist.

Word	**Term Frequency**	**Document Frequency**	**Adjacent Frequency**	**Keyword Frequency**
the	8602	980	2011	3
of	5508	942	2046	115
and	3686	919	1713	38
a	3554	898	2496	3
to	2896	873	488	10
in	2532	829	1027	10
is	2014	749	511	0
for	1889	755	898	12
with	1161	605	531	5
that	1108	590	173	0

Table 3: Top 10 stop words

Using the four metrics, six stoplists were created (see Table 4. The first three select words for the stoplist by term frequency (TF), and three which select words by term frequency excluding those that their keyword frequency is higher than the adjacent frequency (KA). In Table 4, the majority of generated stoplist sizes closely align with their counterparts in [2], except for the stoplists KA($df > 10$) and KA($df > 25$) which both use the adjacent frequency metric. This discrepancy underscores the inherent ambiguity in the creation of such stoplists.

Stoplist	Generated Stoplist Size	Stoplist in [2]
TF ($df > 10$)	1294	1347
TF ($df > 25$)	501	527
TF ($df > 50$)	193	205
KA ($df > 10$)	477	763
KA ($df > 25$)	231	325
KA ($df > 50$)	121	147

Table 4: Comparing the six stoplist sizes generated using the four metrics with those in [2]

4 Results

Each of the stoplists created using the four metrics were set as an input to RAKE, which was run on the 500 scientific documents from the testing set. To evaluate RAKE, the statistical measures precision, recall and F-measure were used. The results of RAKE for every stoplist are listed in Table 5. Table 5 includes the size of the stoplist, the number of keywords extracted by RAKE, the correct keywords (the correct matches between the extracted keywords and the manually assigned keywords), and the statistical measures. In the extracted keywords list, there were duplicate keywords (usually keywords consisting of a single word) that were removed. Additionally, the number of extracted keywords for each document was reduced by 50% as it was observed that keywords with the highest scores were often multi-word phrases, while many keywords with low scores were single words with limited meaning. For each stoplist, we evaluate the document frequency (*df*) by imposing frequency bounds on the occurrence of stop words. The final row of the table represents the execution of RAKE with the default stoplist used by the *nltk* library in Python.

Table 5: Comparison of RAKE performance using stoplists based on term frequency (TF) and keyword adjacency (KA) for a testing dataset of 500 Computer Science abstracts. The table lists the extracted keywords (EK), the correct keywords (CK), and three statistical measures

Stoplist	**Size**	**EK**	**CK**	**Precision**	**Recall**	**F-measure**
TF ($df > 10$)	1294	4870	595	0.12	0.09	0.10
TF ($df > 25$)	501	6822	1031	0.15	0.15	0.15
TF ($df > 50$)	193	8210	1488	0.18	0.22	0.19
KA ($df > 10$)	477	8110	2133	0.26	0.31	0.28
KA ($df > 25$)	231	8527	6752	0.24	0.30	0.27
KA ($df > 50$)	121	8912	6752	0.22	0.29	0.25
Default	179	9307	1854	0.19	0.27	0.23

Upon initial inspection, the statistical measures reveal low values, suggesting sub-optimal performance by RAKE. However, it is essential to note that the choice of stoplist generation method significantly impacts the performance of RAKE. In Table 5, results indicate that the KA ($df > 10$) stoplist outperforms other stoplists,

while the first stoplist in the table exhibits the least favorable performance. Generally, KA stoplists prove more effective in enhancing performance. As mentioned earlier, generating a stoplist solely based on term frequency may exclude meaningful words from the analysis. Moreover, words adjacent to manually assigned keywords and frequently occurring words within keywords are less likely to carry significant meaning.
The outcomes of the analysis can also be attributed to the following factors:

- The corpus comprises abstracts with their respective titles, presenting a challenge for accurate keyword extraction given the constrained context. Access to the entire scientific document, encompassing sections like introduction and related work, would likely yield superior results by providing a more comprehensive context for keyword extraction.

- Authors might employ diverse approaches when defining keywords. Some may choose relevant keywords that appear infrequently in the text, while others might choose a minimal number of keywords, and some may use an excessive amount.

- The size of the training set has a direct impact on stoplist creation, influencing RAKE's performance. Utilizing a larger training set for stoplist generation has the potential to enhance overall performance.

- The approach used to create stoplists significantly influences RAKE's performance. As shown in Table 5, the three KA stoplists exhibit higher values for the measures compared to the TF stoplists.

In [2], superior results were achieved. However, it's noteworthy that there were variations on the results of the generated stoplists, and their approach involved limiting the number of extracted words to one-third of the words in the graph created by TextRank, as outlined in [1].

5 Conclusions

This research investigates the use of RAKE for automatic keyword extraction from scientific documents, focusing on abstracts and their corresponding keywords. Although RAKE demonstrated suboptimal performance, the study provides a detailed analysis of its performance factors, introduces intriguing algorithmic characteristics, describes the stoplist generation methodology, and attempts to replicate the original work presented in [2].

References

[1] R. Mihalcea and P. Tarau. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 404–411, 2004.

[2] S. Rose, D. Engel, N. Cramer, and W. Cowley. Automatic keyword extraction from individual documents. *Text mining: applications and theory*, pages 1–20, 2010.