Comparison of Three Algorithms for Automatic Keyword Extraction*

Mayra A. Paredes-Farrera and Hiram Calvo

Center for Computing Research, National Polytechnic Institute,
Av. Juan de Dios Bátiz s/n, esq. Av. Mendizábal, México, D.F., 07738. México
mparedesf0000@ipn.mx, hcalvo@cic.ipn.mx, www.hiramcalvo.com

Abstract: In this work, we present a comparative analysis of three algorithms for automatic keyword extraction: The first algorithm consists on simple frequency n-gram counts from the text; the second one uses Maximal Frequent Sequences to find relevant words, and the third one is an algorithm based on the complex network model: A complex network is built from bigrams in a text, then the density coefficients of this network for each node (word) are calculated. The relevant words are considered those with a higher density coefficient against a given threshold. The evaluation method used for comparing these algorithms is coverage, as defined in the Document Understanding Conferences (DUC). This coverage measure is determined by the number of overlapping words between the words selected as relevant by each algorithm and a summary model provided by humans (300 documents from DUC 2001). Our experiments show that the algorithm with better performance was Maximal Frequent Sequences.

Keywords: Automatic keyword extraction, Complex Networks, Maximal Frequent Sequences.

1 Introduction

Nowadays, the number of documents available electronically has grown considerably; searching, retrieving or indexing information in such a huge amount of text becomes then a complex task. The usage of keywords is an aid to make these tasks simple and fast.

Keywords within a document are relevant words that provide information about the content or the subject of such text. These words allow people to find in the pool of documents the information they are searching for. Search engines benefit also from the usage of keywords. When keywords are used, it is possible to obtain finer results in shorter time. A keyword can be a unique word (*bodies*), a morphologic root or stem (*bodi*), a lemma (*body*) or even a phrase (*The human body*, *the network*).

Extracting keywords manually might seem a trivial task if the number of documents is small, but for hundreds of documents, this task becomes tedious and costly.

* Work done under partial support of CONACyT, IPN (PIFI, SIP, COFAA), and SNI, Mexico.

Advances in Artificial Intelligence and Applications
Research in Computer Science 32, 2007, pp. 297–304

Received 20/06/07
Accepted 31/08/07
Final version 21/10/07

Because of this, it is convenient to create algorithms to allow the extraction of these words automatically.

There are several techniques that implement the extraction of keywords in different ways. Usually implemented techniques are based on frequencies—count of the number of repetitions of words [9] . In some cases the count of frequencies is enough to fulfill some particular goal [8, 11, 17], but for certain task a more complex techniques are needed [10, 15].

Most of the recent techniques for automatic keyword extraction such as [7, 6 and 14] preprocess the text before extracting the keywords. This pre-processing consists of eliminating stop-words and phrases that begin or end with a stop-word; defining a set of PoS tag sequences; extracting all the words or sequences of words that match any of these; and even marking NP-chunks in the text. We are interested in performing automatic keyword extraction without the need of text pre-filtering, as this latter task would require resources which are language dependent.

In this work, we make a comparison between three techniques for automatic keyword extraction (extracting only one-word keywords). These techniques are based on n-grams (NGs), Maximal Frequent Sequences (MFS) [5] and Complex Network Models (CNM) [16].

Roughly, in NGs, all 2-grams 3-grams, up to 7-grams are extracted from the text. The obtained n-grams are ordered according to its frequency, and subsequently a threshold is used to decide which words are relevant.

For the algorithm based on MFS's, the algorithms first extract sequences from 2 to 7 n-grams from existing words in the text. To determine the relevant words, we define a minimum frequency threshold σ that each sequence in the document must surpass.

The CNM is based on concepts of complex networks following the notion that despite of its complexity, the human language can be represented with the use of graphs (networks). The words in human language interact in sentences and the co-ocurrence of words in sentences reflects language organization in a subtle manner that can be described in terms of a graph of word interactions [16]. CNM uses clustering coefficients of each word and extracts the relevant words that have a coefficient that exceeds a defined threshold.

The remaining of this paper is organized as follows: the next sections explain the algorithms to be evaluated: sections 2 to 4 describe the NGs, MSF and the CMN algorithms respectively, followed by the results of our experiments in Section 5. Finally, conclusions and future work are presented in Section 6.

2 N-Grams Algorithm

An n-gram is a contiguous sub-sequence of *n* items or elements from a given sequence. The underlying idea of the N-grams model (NGs) applied to text is to obtain all sequences of adjacent words (given a window size) which appear together within a text. Punctuation is not considered. A frequent sequence of words contains mostly relevant words for that particular text.

The window size of n-grams can be an arbitrary number ranging from 2 up to the size of the text. For n=2 we have bigrams, which are sequences of two adjacent

words; trigrams are sequences of three words and so on. In Figure 1 we show an example of the extraction of n-grams corresponding for an example text.

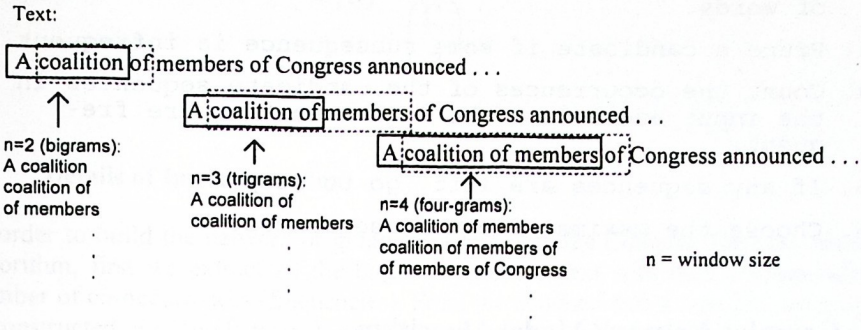

Figure 1. Example of adjacent n-grams extraction

3 Maximal Frequent Sequences Algorithm

For our purposes, we consider a Maximal Frequent Sequence (MFS) as a chain of words that appear often and are not contained in any other longer frequent sequence in a document collection. A sequence is considered to be frequent if it appears in at least σ documents, when σ is the frequency threshold given [2]. MFS are described by the following definitions:

Assume S is a set of documents, and each document consists of a sequence of words.

Definition 1 *A sequence* $p = a_1 \ldots a_k$ *is a subsequence of a sequence q if all the items* a_i, $1 \leq i \leq k$ *occur in q and they occur in the same order as in p. If a sequence p is a subsequence of a sequence q, we also say that p occurs in q.*

Definition 2 *A sequence p is frequent in S if p is a subsequence of at least σ documents of S, where σ is a frequency threshold given.*

Definition 3 *A sequence p is a maximal frequent (sub) sequence in S if there does not exist any other sequence p' in S such that p is a subsequence of p' and p' is frequent in S.*

The pseudocode of the Bottom-up algorithm [12] (taken from [1]) that we implemented to obtain the MFS is shown in Figure 2.

```
Input: A set of sequences, a frequency threshold
Output: A set of maximal frequent sequences

1. Collect all items of the input sequences, count them
   and select the frequent ones.
```

2. Build candidates sequences of length k+1 from frequent sequences of length k, where k is the number of words.
3. Prune a candidate if some subsequence is infrequent.
4. Count the occurrences of the candidate sequences in the input and select the sequences that are frequent.
5. If any sequences are left, go to Step 2.
6. Choose the maximal frequent sequences.

4 Complex Network Model Algorithm

The Complex Network Model (CNM) algorithm is based on concepts of complex networks and small-worlds. A complex network is a network (graph) with a non-trivial topological structure; the complex network built from a text presents small-world's (SW) characteristics The SW pattern can be detected from the analysis of two basic statistical properties: the clustering coefficient C and the path length d [16]. Antiqueira *et al.* [3 and 4] have modeled the text as a complex network. They show that it is possible to use this model to find solutions to diverse problems for natural language processing tasks.

In particular, TextRank [14] uses graphs for keywords and sentence extraction. The model we present here is similar to TextRank, but it is simpler in the sense that we do not consider the damping factor which has the role of integrating into the model the probability of jumping from a given vertex to another random vertex in the graph. In addition, we are not preprocessing the text (stop-word filtering, PoS tagging) because doing so would require language-dependent resources while we are searching for general language algorithms.

The graph for the language in a text (Ω_L), can be defined by $\Omega_L = (W_L, E_L)$, where $W_L=\{w_i\}$, $(i=1,\ldots, N_L)$ is the set of N_L words, and $E_L=\{\{ w_i, w_j \}\}$ is the set of edges or connections between words. $\xi_{ij} = \{w_i, w_j\}$, $(i,j=1,\ldots, N_L)$ indicates that there is an edge between words w_i and w_j, where $\xi_{ij}=1$ if a link exists and 0 otherwise. An edge is defined in this case as the existence of a bigram with frequency > 0 for the words w_i and w_j. The number of links per word is k. The set of nearest neighbors of a word $w_i \in W_L$ as defined by $\Gamma_i=\{j \mid \xi_{ij}=1\}$.

For the purpose of finding relevant words we calculated the clustering coefficient for each word. The clustering coefficient is defined as the number of connections between the words $w_i \in W_L$ as defined in equations (1) and (2).

$$L_i = \sum_{j=1}^{N_L} \xi_{ij} \left[\sum_{k \in \Gamma_i; j<k} \xi_{jk} \right] \quad (1)$$

$$C_v(i) = \frac{L_i}{\binom{\Gamma_i}{2}} \tag{2}$$

4.1 Details of Implementation

In order to build the network or graph for a text with the Complex Network Model algorithm, first we extract all the bi-grams from the text with their corresponding number of co-occurrences (frequencies). From the obtained data a non-directed graph is constructed. Each node from this graph represents a word, whereas every link of the node represents the adjacency of two words from each bigram. Each link is weighted by the number of times that the corresponding bi-gram appears in the text.

The clustering coefficient is calculated for each node (word) in the network and sorted in reverse order as show in Figure 4. Words with a clustering coefficient greater than 0 are considered relevant.

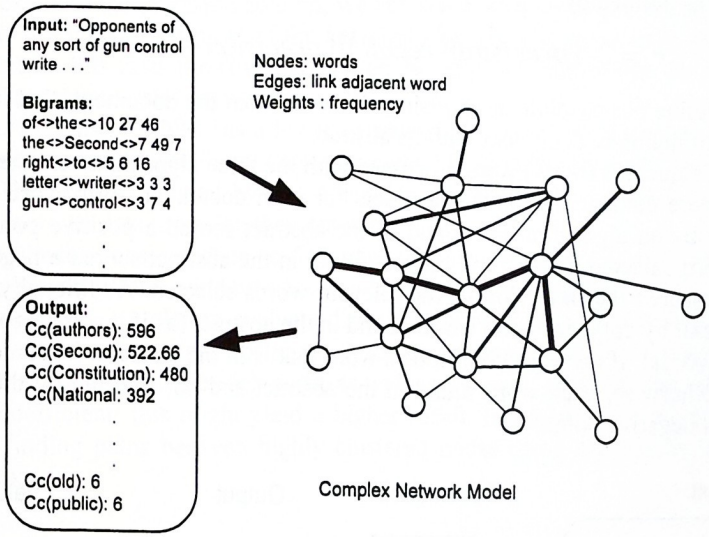

Fig. 2. Diagram of the CNM implementation. Darker links show a greater frequency of occurrence for particular bigrams in a text. *Cc*(*word*) is the clustering coefficient of *word*.

5 Experiments

We experimented with the available 300 documents from the Document Understanding Conference 2001 (DUC). The DUC is a series of evaluations of automatic text summarization systems organized by the National Institute of Standards and Technol-

ogy (NIST). Each document (body) has a summary model (abstract) written by humans; the documents topics are varied, as well as the size of each document. The average length of the documents is 762 words, ranging from 4580 words to 138 words.

Evaluation measure: The evaluation method we used for comparing these algorithms is *coverage*, as defined in DUC. This *coverage* measure is determined by the number of overlapping words between the relevant words chosen by each algorithm, and the abstract of each document. We rely on the idea that abstracts contain the most relevant keywords for a text. Precision is defined as a measure of the proportion of selected items that the system got right:

$$precision = \frac{tp}{tp + fp} \tag{3}$$

Recall is defined as the proportion of the target items that the system selected:

$$recall = \frac{tp}{tp + fn} \tag{4}$$

F-measure is a single measure of overall performance, combine precision and recall and as defined by:

$$F = 2 \cdot (precision \cdot recall)/(precision + recall) \tag{5}$$

We define our baseline as choosing all words from the document; that is, the intersection of the body document with its abstract.

The documents (body) were processed with the three algorithms and the results obtained were compared with the abstracts for each document. Each word selected as relevant by an algorithm and found in the abstract scored a positive point; whereas each word selected as relevant and not found in the abstract scored a negative point. This measured the **precision** of the relevant words selected. Additionally, we measured **recall** by comparing the words found in the abstract (as if it were a gold-standard for keywords). If the algorithm finds a word that is in the abstract, it scored a positive point. Otherwise, each word found in the abstract and not selected by the algorithm scored a negative point.

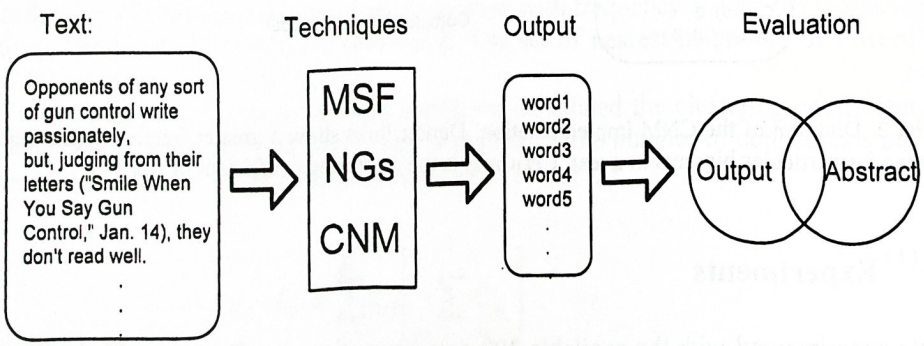

Figure 3. Experiment steps.

In Table 1, we show the results of our experiments obtained applying each one of the three algorithms to the DUC documents.

Table 1. Results of our experiments with the three algorithms.

Algorithm	Precision	Recall	F-measure
Body (baseline)	38.08	74.31	49.92
MFS	60.34	24.85	**34.16**
CNM	**62.92**	13.09	20.48
NGs	43.09	**25.01**	31.17

6 Conclusions and Future Work

We have presented a comparison between three techniques for automatic keyword extraction. Our results show that the algorithm with better performance considering both precision and recall was MFS. The baseline has the highest f-measure. It might seem that none of the algorithms was able to perform better than the baseline, however if we examine the precision column, we can see that the baseline has a very low precision. It is expected for most of the keywords from the abstract to be present in the body text. In this case, the coverage is high, but this also means that the keyword set contains many non-relevant words.

The algorithm using CNM has a lower combined performance; however it has the highest precision of all algorithms. This is because this algorithm obtains few but precise data. The results obtained by this algorithm are useful for certain applications which rely more on data precision than quantity.

Our experiment considered up to 7-grams. We shall experiment with greater n-gram sizes, as well greater window sizes to allow gaps in the n-grams.

As a future work we plan to evaluate directly with professional indexer keywords instead of comparing against abstracts. Another improvement is to consider the neighbors that are close to the each node, as well as the node itself for calculating the clustering coefficient; this might yield a higher recall. In addition we might experiment with finding paths between highly clustered nodes using algorithms based on random walks.

References

1. Ahonen-Myka, Helena, and Antoine Doucet, Data mining meets Collocations discovery. In Inquiries into Words, Constrints and Contexts, Festschrift for Kimmo Koskenniemi, pp. 194–203. CLSI Publications, University of Stanford, 2005.
2. Ahonen-Myka, Helena. Finding all maximal frequent sequences in text. In ICML99 Workshop, Machine Learning in Text Data Analysis. Bled, Slovenia, 1999.
3. Antiqueira L., M.G.V. Nunez, O.N Oliveira Jr, L.F Costa, Modelando Textos como Redes Complejas. In Anais do III Workshop em Tecnologia da informação e da Linguagem humana. TIL São Leopoldo-RS Brazil. July 22-26, 2005.

4. Antiqueira L., M.G.V. Nunez, O.N Oliveira Jr, L.F Costa. Strong correlations between text quality and complex networks features. Physica A, Volume 373, p. 811-820, 2007.
5. García-Hernández, René A., Jose Fco Martínez-Trinidad, Jesús Ariel Carrasco-Ochoa. A Fast Algorithm to Find All the Maximal Frequent Sequences in a Text. CIARP 2004, pp. 478-486, 2004.
6. Hult, A., Combining Machine Learning and Natural Language Processing for Automatic Keyword Extraction. Ph.D. Thesis, Departament of Computer and Systems Sciences. Stockholm University, April 2004.
7. Hult, A., Improved automatic keyword extraction given more linguistic knowledge. In Proceedings of the 2003 Conference on empirical Methods in Natural Language Processing, Japan, August, 2003.
8. Justeson, John S., and Slava M Katz. Co-ocurrences of antonymous adjectives and their contexts. Computational Linguistics 17:1-19, 1991.
9. Justeson, John S., and Slava M. Katz. Technical terminology: some linguistic properties and an algorithm for identification in text. Natural Language Enginnering 1:9-27, 1995.
10. Kovács László and Helena Ahonen-Myka. Algorithm for Maximal Frequent Sequences in Document Clustering. 3rd International Symposium of Hungarian Researchers on Computational Intelligence. 2002.
11. Kupiec, Julian, Jan Pedersen, and Francine Chen. A trainable document summarizer. In SIGR '95, pp. 68-73, 1995.
12. Mannila Heikki, Toivonen Hannu, Inkeri Verkamo Discovering frequent episodes in sequences. In Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD'95), pages 210-215. Montreal, Canada. 1995.
13. Manning, Christhopher D., Shütze Hinrich. Foundations of Statistical Natural Language Processing. MIT Press, 2000.
14. Mihalcea Rada, Paul Tarau. TextRank: Bringing Order into Texts. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2004), Barcelona, Spain, July 2004.
15. Nguyen, Son N., Sun Xingzhi and Orlowska Maria E. Improvements of IncSpan: Incremental Mining of Sequential Patterns in Large Database. PAKDD 2005, pp. 442-451. 2005.
16. Ramon Ferrer I Cancho and Ricard V. Solé, The small world of human language, Proceedings of The Royal Society of London. Series B, Biological Sciences, 268(1482):2261–2265, 2001.
17. Ross, Ian C., and Jonh W. Tukey. Introduction to these volumes. In John Wilder Tukey (ed.), Index to Statistics and Probability, pp iv-x. Los Altos, CA:R & D Press, 1975.
18. Satanjeev Banerjee and Pedersen Ted. The Design, Implementation and Use of Ngram Statistics Package. In Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics, February 17-21, 2003.
19. Thiago Alexandre Salgueiro Pardo, Lucas Antiqueira Maria das Graças Volpe Nunes, Osvaldo N. Oliveira Jr. Luciano da Fontoura Costa "Using Complex Networks for Language Processing: The case of Summary Evaluation", 2006.