Naïve Algorithms for Keyphrase Extraction and Text Summarization from a Single Document Inspired by the Protein Biosynthesis Process

Daniel Gayo-Avello, Darío Álvarez-Gutiérrez, and José Gayo-Avello

Department of Informatics, University of Oviedo, Calvo Sotelo s/n 33007 Oviedo (SPAIN)
Tel. +34 985 10 50 94 / Fax +34 985 10 33 54
dani@lsi.uniovi.es

Abstract. Keywords are a simple way of describing a document, giving the reader some clues about its contents. However, sometimes they only categorize the text into a topic being more useful a summary. Keywords and abstracts are common in scientific and technical literature but most of the documents available (e.g., web pages) lack such help, so automatic keyword extraction and summarization tools are fundamental to fight against the "information overload" and improve the users' experience. Therefore, this paper describes a new technique to obtain keyphrases and summaries from a single document. With this technique, inspired by the process of protein biosynthesis, a sort of "document DNA" can be extracted and translated into a "significance protein" which both produces a set of keyphrases and acts on the document highlighting the most relevant passages. These ideas have been implemented into a prototype, publicly available in the Web, which has obtained really promising results.

1 Introduction

As the saying goes, *"Time is Money"*, *"Information is Power"*. So, most of us want to earn the most the power (i.e., information or knowledge) at the lowest possible cost (i.e., as soon as possible and with relatively little effort). To accomplish this, many communities make use of guidelines to write "easy-to-read" documents. These guidelines can be as simple as attaching an abstract and/or a list of keywords at the beginning of each document.

However, most of the documents available on a daily-basis have neither abstract nor keywords. Examples of these "time consuming" documents are e-mail messages, web pages, newsgroup posts, etc. On the other hand, such documents are provided by digital means, so, at least, they are suitable for automatic processing. In fact, there are plenty and very different Natural Language Processing (NLP) techniques to help us to sort through this information overload (e.g., language identification [1][2], document clustering [3][4], keyword extraction [5][6] or text summarization [7]).

Some of these techniques require human supervision [5] while others don't [1-4][6][7]. Several don't require training [6] but others do [1-5][7]. Some rely only on statistical information [1-4][6] and others employ complex linguistic data [5][7]. A few use only one document [6] while others need a document corpus [1-5][7].

The question becomes wouldn't it be great to use only one technique to carry out several of these tasks? Ideally, it should be extremely simple (i.e., it should rely only on free text instead of linguistic data), fully automatic (i.e., it should need neither human supervision nor *ad hoc* heuristics) and scalable (i.e., feasible with both single documents –a web page– and document corpora –web sites).

Biology has inspired many computational techniques that have proven feasible and reliable (e.g., genetic algorithms or neural and immune networks). So, trying to find such a technique we also turned to biology. Among living beings each individual is defined by its genome, which is composed of chromosomes, which are divided into genes and then constructed upon genetic bases. Likewise, if we consider a document as an individual from a population –a document corpus– we can see that documents are composed of passages, divided into sentences built upon words. Following this analogy, we hypothesized that two documents written in the same language or semantically related would show similar "document genomes". This paper will explain how these document genomes can be extracted and translated into "significance proteins" (i.e., keywords, keyphrases and summaries).

2 Biological Definitions

Since our proposal is heavily inspired by biological phenomena some definitions are provided to clarify various aspects of the techniques and algorithms described later in this paper.

Definition 1: Nitrogenous bases
Molecules involved in the pairing up of DNA and RNA strands. They include adenine (A), thymine (T), cytosine (C), guanine (G) and uracil (U). Uracil only exists in RNA replacing thymine which is only present in DNA. Adenine, cytosine and guanine are common to both DNA and RNA. The possible base pairs are AT or AU and GC.

Definition 2: Nucleotide
A nucleotide is an organic molecule composed of a nitrogenous base, a pentose sugar (deoxyribose in DNA or ribose in RNA), and a phosphate or polyphosphate group. Nucleotides are the monomers of nucleic acids.

Definition 3: Deoxyribonucleic acid (DNA)
DNA is a polymer and the main chemical component of chromosomes. It is usually the basis of heredity because parents transmit copied portions of their own DNA to offspring during reproduction propagating their traits. DNA is a pair of chains of nucleotides entwined into a "double helix". In this double helix, two strands of DNA come together through complementary pairing of the nucleotides' bases, because of this, DNA is usually represented as a unique text string (e.g. ...`GGCGATACATG`...) which has been of primary importance for the development of bioinformatics.

Definition 4: Ribonucleic acid (RNA)
RNA is, as DNA, a nucleic acid although slightly different from DNA, both in structure and function. As it has been explained above, RNA is composed of the same bases as DNA except for uracil. The RNA structural differences give the molecule greater catalytic versatility and help it to perform its many roles in the transmission of genetic information from DNA (by transcription) and into protein (by translation).

Definition 5: Transcription
The process of transcribing genetic information from DNA into a messenger RNA molecule using the DNA molecule as a template. Transcription is the prior step before protein biosynthesis.

Definition 6: Messenger RNA (mRNA)
mRNA is transcribed directly from a gene's DNA and carries the code for a particular protein from the nucleus to a ribosome in the cytoplasm and acts as a template for the formation of that protein. mRNA is a single-stranded molecule.

Definition 7: Transfer RNA (tRNA)
A relatively small RNA molecule that transfers a particular amino acid to a growing polypeptide chain at the ribosomal site during translation.

Definition 8: Amino acid
Amino acids are the constituent molecules of proteins. There are 20 amino acids directly expressed by means of DNA. However, a protein can contain amino acids that differ from these twenty; if this is the case, the different amino acid has been transformed after translation.

Definition 9: Protein
Proteins are polymers consisting of one or more strings of amino acids. Each string folds into a 3D structure, existing of four different levels of protein structure. However, for our purposes, we are interested in only two of them: (1) Primary structure: the linear amino acid sequence forming a polypeptide. (2) Quaternary structure: the association of multiple polypeptide subunits to form a functional protein. The primary structure is made during the translation and the higher structures are reached during the protein folding process. Proteins are primary components of living organisms; they can be used as hormones, enzymes, structural elements or even to obtain energy.

Definition 10: Ribosome
A ribosome is a structure composed of RNA (ribosomal RNA or rRNA) and proteins that can translate mRNA into a polypeptide chain (usually a protein). Ribosomes are found in the cytoplasm of all cells and consist of two subunits.

Definition 11: Translation or protein synthesis
Protein synthesis involves three steps: (1) preparing tRNA molecules for use by the ribosome; (2) attaching the ribosome to the mRNA; and (3) the initiation, elongation and termination phases of translation, where an amino acid chain forming the primary structure of a protein is constructed. The process of translation will be described with more detail in the next section.

Definition 12: Bioinformatics
Information technology as applied to life sciences. For instance, the techniques used for the collection, storage, retrieval, data mining and analysis of genomic data. Other applications include sequence alignment, protein structure prediction, etc.

3 Synthesis, Folding, and Functions of Proteins

As defined above, DNA is capable of encoding 20 different types of amino acids; these are the basic components of proteins which play essential roles in almost every biological process. Therefore, cells need to produce proteins using their DNA as a kind of "blueprint".

However, DNA is not very chemically versatile and, moreover, too valuable to work directly on top of it to produce the needed proteins. Because of this, to synthesize any protein the cell must first copy a portion of its DNA (i.e., the gene encoding the protein) into a single-stranded molecule of mRNA which is sent to the cytoplasm. There, it will be used by ribosomes as a template to compose the final protein. This prior step is called transcription while the process to obtain a protein from the mRNA molecule is called protein synthesis or translation.

As it will be shown later in this paper, our proposal is freely inspired by the information encoding capabilities of DNA, the transcription of DNA into mRNA, its translation into the primary structure of a protein and the folding of this protein to reach its fully functional form. Therefore, the following paragraphs offer a more thorough description of the translation and folding processes.

3.1 The Translation Process

Protein synthesis begins with the attachment of the small ribosomal subunit to the mRNA string. Then the initiator tRNA molecule binds to the start codon[1] AUG. This step is named initiation (see Fig. 1) and after it begins the elongation phase.

The elongation phase starts when the large ribosomal subunit is attracted by the initiator tRNA and binds to the small subunit completing the ribosome (see Fig. 2). During this phase the ribosome moves along the mRNA string one codon at a time. Each of these codons in the mRNA has a complementary anticodon in tRNA molecules which, as we already know, carry amino acids. This way the polypeptide

[1] A codon is a sequence of three adjacent nucleotides in mRNA determining the binding of a particular amino acid (carried by a tRNA molecule) or the signal to stop the translation.

sequence, dictated by DNA and transmitted by mRNA continues growing (see Fig. 3) until the stop codon (UAA, UAG or UGA) is reached by the ribosome. At that moment the translation process enters the termination phase.

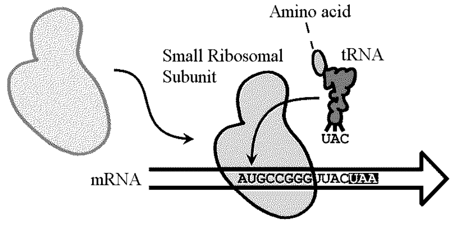


Fig. 1. Initiation of the translation process.

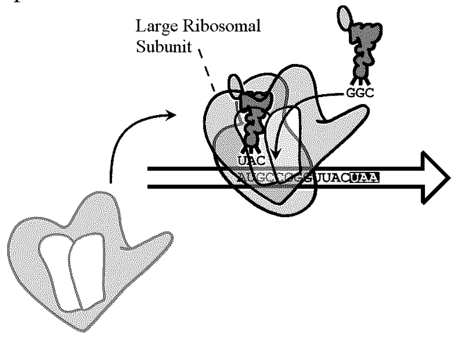


Fig. 2. Start of the elongation phase.

When the ribosome reaches the stop codon no tRNA is attracted, the ribosome dissociates and both ribosomal subunits leave the mRNA. The product of this process is a polypeptide, that is, the primary structure of a protein (see Fig. 4).

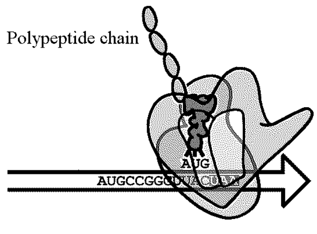


Fig. 3. End of the elongation phase.

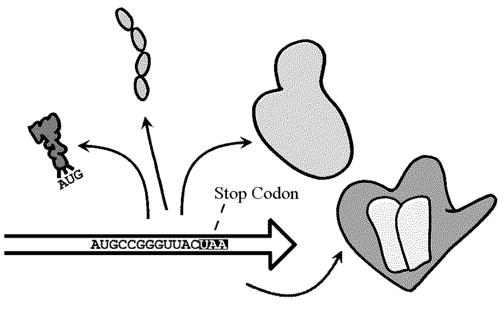


Fig. 4. Termination of the translation process.

3.2 Protein Folding and Functions

As it was explained in Definition 9, proteins must fold into a 3D structure to perform their many functions. All of the information required to reach the final folded form is contained in the primary structure since proteins fold into low energy configurations depending on the interactions between their constituent amino acids. Predicting protein folding from the amino acid sequence has been a major problem for over a decade and many techniques have been proposed (e.g., LINUS [8] or folding@home[2][9]).

[2] http://www.stanford.edu/group/pandegroup/folding

Many of the ideas behind these techniques such as maximization of entropy during the folding process [8] have greatly influenced some of the algorithms described later in this paper. The functions of some proteins, such as hormones and enzymes, were also especially influential since we were interested not only in obtaining “document proteins” (i.e., keyphrases) but also in the possibility of such “proteins” having active effects on the original document to achieve automatic summaries.

4 A DNA for Natural Language?

Now that we have broadly described the main biological phenomena in which our techniques have rooted in, we can more clearly define our proposal. Let’s start with the concept of “document genome”. To see what a “document genome” looks like let’s consider an extremely short text shown in Fig. 5.

```
The rain in Spain stays
  mainly in the plain.
```

Fig. 5. An example document.

```
    ain-(3) -in-(2) … he-p(1)
e-pl(1) -pla(1) plai(1) lain(1)
```

Fig. 6. Partial list of 4-grams for the previous document. It is reverse-ordered by simple frequency and blanks have been replaced by hyphens.

A common technique to analyze texts is based on the use of *n*-grams which are simply sequences of length *n*. They can include either words or characters and the items do not need to be contiguous. However, frequently the term *n*-gram refers to slices of adjoining *n* characters. For the purposes of this paper we will use this definition of *n*-gram throughout. Moreover, while it is common when working with *n*-grams to obtain a variety of different lengths, our document DNA will use fixed length *n*-grams[3], usually 4-grams or 5-grams (see Fig. 6).

By comparing the “most frequent” *n*-grams in two different sequences it is possible to perform language identification and, to some extent, document categorization. However, these “most frequent” *n*-grams are usually obtained by inspection [3].

Therefore, *n*-grams have commonly been used to perform basic analysis of natural language and are a feasible way of performing language identification. However, these lists of *n*-grams with their corresponding frequency are quite distinct from DNA in living beings. On the other hand, since we are working with fixed length *n*-grams we can easily construct a string including all the *n*-grams from the document, repeating each *n*-gram as many times as specified by its simple frequency and ordering them alphabetically (see Fig. 7).

This form of pseudo-DNA is well suited to perform comparisons among two or more documents’ genomes. The very same sequence alignment algorithms employed in bioinformatics can be used over these pseudo-DNA strings to perform language

[3] By doing this the algorithms are much simpler. It would be interesting to study the feasibility of mixing *n*-grams of different sizes into a single pseudo-DNA string.

identification and document clusterization and, in fact, these algorithms can be greatly simplified taking advantage of the alphabetical ordering of the "genes".

`**ain-ain-ain-**ainl**ays-**e-pl**e-ra**he-p**he-r**-in--in-**in-i**in-S**in-s**`
`in-t**inly**lain**ly-i**-mai**main**n-in**n-Sp**n-st**n-th**nly-**pain**-pla**plai**`
`-rai**rain**s-ma**-Spa**Spai**-sta**stay**tays**The-**the-**-The**-the**y-in**ys-m**`

Fig. 7. Example document genome. Bold type is used only for the sake of clarity.

While this technique appears to be equivalent to the "out-of-place" measure used in [3] to compare *n*-gram profiles the concept of document distance is, perhaps, more easily understood using the pseudo-DNA and sequence alignment algorithms. Currently the application of this technique to language identification and topic clustering is a work in progress, but it has many more possible applications such as keyphrase extraction and text summarization.

This pseudo-DNA needs, however, some improvements. Since the simple frequency depends on the length of the document we cannot use it to determine the number of repetitions of each *n*-gram. Instead, we must use their relative frequency and perform some scaling to transform the floating point values into integers. Also, using a logarithmic scale appears to be an adequate solution. What is more, the simple frequency is not the most relevant measure associated with each *n*-gram. For instance, following with the prior example document, we cannot say that `Spain` and `mainly` are equally relevant although the 4-grams `Spai` and `inly` share the same frequency.

Working with bigrams there are various measures to show the relation between both elements of a pair (e.g., mutual information, Dice coefficient, ϕ^2 coefficient, Loglike, etc.) Such measures provide much more interesting information than simple frequency but they cannot be applied straight forward to *n*-grams without a generalization. For this proposal we have chosen the Fair Specific Mutual Information (*SI_f*) [10] (see equations 1 and 2).

$$SI\_f((w_1 \ldots w_n)) = \log\left(\frac{p(w_1 \ldots w_n)}{Avp}\right) \tag{1}$$

$$Avp = \frac{1}{n-1} \cdot \sum_{i=1}^{n-1} p(w_1 \ldots w_i) \cdot p(w_{i+1} \ldots w_n) \tag{2}$$

In this measure, $w_1 \ldots w_n$ represents an *n*-gram, while $w_1 \ldots w_i$ and $w_{i+1} \ldots w_n$ are two fragments of that *n*-gram –e.g., `Spai`, `S` and `pai`, respectively. In addition $p(w_1 \ldots w_n)$ is the probability (i.e., relative frequency) of the full *n*-gram while $p(w_1 \ldots w_i)$ and $p(w_{i+1} \ldots w_n)$ are the appearance probabilities of each segment not only as part of the original *n*-gram[4].

[4] For instance, in the case of the 4-gram `inly`, the segment `in` appears as left-sider in five 4-grams: `[in-S]pain`, `ra[in-i]n`, `Spa[in-s]tays`, `[in-t]he` and `ma[inly]`.

A two-step algorithm is provided later in this paper (see Appendix). The first step, *precalculateMatrix*, precalculates a table of data from a sequence of *n*-grams. The second one uses that table to calculate the *SI_f* for a particular *n*-gram. For instance, by applying these algorithms we found that the significances for `Spai` and `inly` are `2.013` and `1.975`, respectively.

However, the values themselves are of little importance. What is really important is that they allow us to rank the *n*-grams according to their relevance. From this ranking the document DNA is built. But, first, the ranking must be scaled to the range 1...K (being K a power of 2). This way the most relevant *n*-gram will appear K times in the pseudo-DNA string while the least relevant will appear only once. As explained above, all *n*-grams are placed in alphabetical order within the pseudo-DNA string. To summarize this section we provide the following definitions.

Definition 13: N-Gram
A sequence of ISO-8859-1 (Latin 1) alphabetic characters, either lowercase or uppercase, or blanks. The length of the sequence (i.e., the size of the *n*-gram) is a parameter of the document DNA since all the *n*-grams within a document DNA must be equal in length. The most common being 4-grams and 5-grams.

Definition 14: Document Gene
A document gene is a variable length repetition of a unique *n*-gram. The number of appearances of the *n*-gram in the gene depends on its ranking within the original document's *n*-gram sequence according to the Fair Specific Mutual Information measure (*SI_f*). A gene has at least one *n*-gram and at most K, a power of 2. K is a parameter of the document DNA.

Definition 15: Document DNA
A document DNA is a text string representing a sequence of document genes in alphabetical order without any separator.

5 Synthesis, Folding, and Effects of Significance Proteins

According to definitions 13 to 15, we can extract from a document written in any western language a "genome" which would encode the "significance" of the different *n*-grams occurring in that document. If this "genome" is in some way similar to the DNA from living organisms it should hide some valuable information within it. Such information could be extracted by means of a translation process similar to the protein biosynthesis. The results from this translation process (i.e., the significance protein) would ideally provide a way to compact themselves in a suitable representation of the most relevant information in the document (i.e., a list of keyphrases) and should also be capable of modifying the document itself (i.e., providing an automatic summary).

5.1 Document's DNA Translation into a Significance Protein

To implement such ideas we must previously find the counterparts for mRNA, tRNA and the ribosome in this scenario. Since the techniques shown in this paper are freely inspired by protein biosynthesis we can redefine some roles. This way, our proposed technique will use the elements shown in Table 1.

Table 1. Biological elements and their computational counterparts.

Biological element	Computational element
mRNA	Document's plain text
tRNA	Spliced document DNA
Polypeptide chain (protein's primary structure)	Document chunks with significance weights
Ribosome	The *ribosomalTranslation* Algorithm
Folded protein	Document's keyphrases

The basic ideas behind this technique are quite straightforward:

1. The document DNA encodes, by means of the different lengths of its genes, the significance of each *n*-gram from the document. This DNA can be spliced into chunks which will carry a "significance weight" attached to a specific *n*-gram behaving in a similar way to tRNA (see Fig. 8).
2. The plain text from the document doesn't provide any information to the computer about the different relevance of each passage and word. However, it is well-suited to be sequentially processed and, thanks to our computational tRNA, a weight can be assigned to each *n*-gram from this text (see Fig. 9).
3. We already know that real proteins fold into a low-energy conformation or, which is the same, a maximum entropy state [8]. In a similar fashion, the "significance weights" assigned to each *n*-gram from the document's text will be accumulated while the mean significance per character continues growing. This way, a sequence of chunks of text with maximum significance will be obtained, being the equivalent to the polypeptide chain in protein biosynthesis.
4. All these tasks will be carried out by an algorithm (i.e., our ribosome) shown in the Appendix.

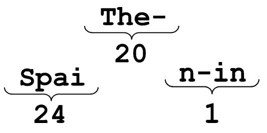


Fig. 8. Document DNA is spliced into "tRNA molecules" carrying significance weights.

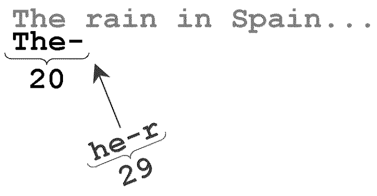


Fig. 9. This tRNA attaches to the document's text endowing it with significance values.

5.2 Significance Protein Folding: Keyphrase Extraction

The *ribosomalTranslation* algorithm splits the document's text into a sequence of chunks with maximum significance. This sequence is analogous to the primary structure of a protein since it has all the information to reach the final functional form, however, it still has to undergo the folding process (see Fig. 10).

This process is performed by the algorithm *proteinFolding* (see Appendix) which depends on three different algorithms: (1) *selfAttract* (see Appendix), (2) *mergeKeys* and (3) *sliceKeys*. The last two algorithms are quite simple: *mergeKeys* receives as input a sequence of weighted keyphrases, a threshold and a window size and merges all the keys weighted above the threshold and inside the specified window. *sliceKeys* simply drops any keyphrase below the specified threshold.

5.3 Significance Protein Effects on the Document: Summarization

Once the *proteinFolding* algorithm has produced the significance protein this can act on the document's text to modify it, producing both a highlighted version of the document and an automatic summary. The algorithm to apply the protein over the document's text, *blindLight*, and another algorithm to obtain a refined list of keyphrases from the folded protein are shown in the Appendix.

```
of-Liberia's(22.06) -for-ECOWAS(21.83) a-Liberia's(21.79)
Scott(21.75)     Economic(21.32)     House-Nyudueh(21.31)
Liberia's(21.25)    National(20.94)    House-Nyudueh(20.91)
A-West(20.87) International(20.82) Taylor's(20.82)…
```

Fig. 10. Top 12 most significant chunks from a web page[5]. Chunks that will likely come together to form the "folded protein" are shown in bold (blanks have been replaced by hyphens).

6 The "blindLight" Prototype

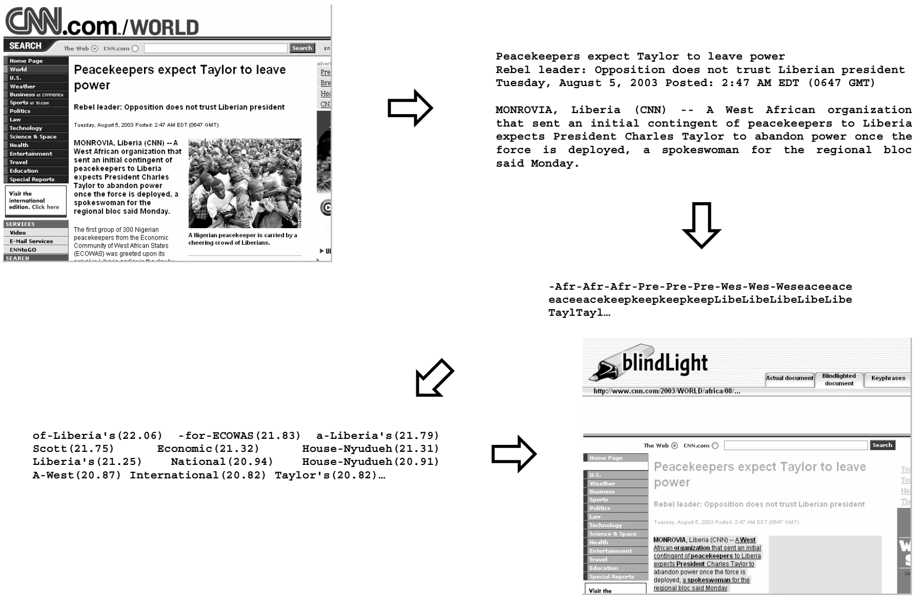
To test the feasibility of previous ideas we developed a prototype and made it publicly available in the Web[6]. This prototype, called *"blindLight"*, receives a URL from the user and returns as results three different views from the original web page: a "blindlighted" version of the page, a list of keyphrases and an automatic summary. To reach such results the prototype performs a number of steps (see Fig. 11) which are described below.

1. The web page is distilled to obtain the main contents as plain text.
2. From these contents two genomic sequences are extracted (a 4-gram gene sequence and a 5-gram gene sequence –see Section 4).

[5] http://edition.cnn.com/2003/WORLD/africa/08/04/liberia.peacekeepers/

[6] http://www.purl.org/NET/blindlight

3. Both sequences are translated into significance proteins (Section 5.1) which, in turn, are folded (Section 5.2) obtaining two lists of candidate keyphrases.
4. The folded significance protein acts on the document obtaining the most relevant passages from the document (Section 5.3).
5. The results are shown to the user (see Fig. 12 and Fig. 13).


CNN.com/WORLD
Peacekeepers expect Taylor to leave power
Rebel leader: Opposition does not trust Liberian president
blindLight
Actual document
Blindlighted document
Keyphrases

```
Peacekeepers expect Taylor to leave power
Rebel leader: Opposition does not trust Liberian president
Tuesday, August 5, 2003 Posted: 2:47 AM EDT (0647 GMT)

MONROVIA, Liberia (CNN) -- A West African organization
that sent an initial contingent of peacekeepers to Liberia
expects President Charles Taylor to abandon power once the
force is deployed, a spokeswoman for the regional bloc
said Monday.
```

```
-Afr-Afr-Afr-Pre-Pre-Pre-Wes-Wes-Weseaceeace
eaceeacekeepkeepkeepkeepLibeLibeLibeLibeLibe
TaylTayl…
```

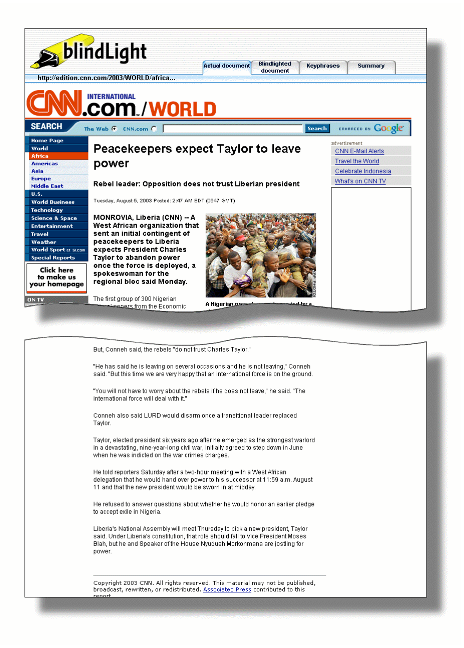
```
of-Liberia's(22.06)  -for-ECOWAS(21.83)  a-Liberia's(21.79)
Scott(21.75)       Economic(21.32)      House-Nyudueh(21.31)
Liberia's(21.25)     National(20.94)    House-Nyudueh(20.91)
A-West(20.87) International(20.82) Taylor's(20.82)…
```

Fig. 11. The different steps to "blindlight" a document.

7 Partial Results and Conclusions

Further and more thorough analysis of these keyphrase extraction and summarization techniques is needed. However, some proof-of-concept tests have been performed and the results, although they could certainly be improved, are extremely promising. This preliminary experiment was carried out over 20 articles from three different journals (*Psycoloquy*, *Behavioral & Brain Sciences Preprint Archive* –both about cognition– and the *Journal of the International Academy of Hospitality Research* –about hotel industry). All of which are available on the Web and include a list of keywords provided by the articles' authors.

The test was quite straightforward. First, the described algorithms were applied over "censored" versions of the papers (i.e., the abstract, list of keywords and references were manually removed from each article) to obtain a list of keyphrases. Then, the percentage of matches was computed for every execution. An automatically extracted keyphrase was a match if it was in the original list of keywords; however, keywords from the list not occurring in the text of the article were not taken into account. Table 2 shows an outline of the results. However, these results have two major caveats: (1) some authors provide many keywords not included in the article[7]; (2) our techniques obtain much more keyphrases apart from the matches so experts from the respective fields should check their relevance to the article.



Peacekeepers expect Taylor to leave power

Rebel leader: Opposition does not trust Liberian president

Tuesday, August 5, 2003 Posted: 2:47 AM EDT (0647 GMT)

MONROVIA, Liberia (CNN) -- A West African organization that sent an initial contingent of peacekeepers to Liberia expects President Charles Taylor to abandon power once the force is deployed, a spokeswoman for the regional bloc said Monday.

The first group of 300 Nigerian peacekeepers from the Economic

A Nigerian peacekeeper is carried by a

But, Conneh said, the rebels "do not trust Charles Taylor."

"He has said he is leaving on several occasions and he is not leaving," Conneh said. "But this time we are very happy that an international force is on the ground.

"You will not have to worry about the rebels if he does not leave," he said. "The international force will deal with it."

Conneh also said LURD would disarm once a transitional leader replaced Taylor.

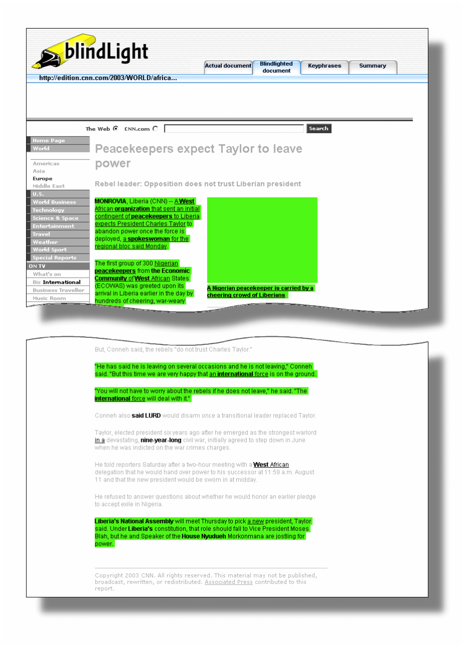
Taylor, elected president six years ago after he emerged as the strongest warlord in a devastating, nine-year-long civil war, initially agreed to step down in June when he was indicted on the war crimes charges.

He told reporters Saturday after a two-hour meeting with a West African delegation that he would hand over power to his successor at 11:59 a.m. August 11 and that the new president would be sworn in at midday.

He refused to answer questions about whether he would honor an earlier pledge to accept exile in Nigeria.

Liberia's National Assembly will meet Thursday to pick a new president, Taylor said. Under Liberia's constitution, that role should fall to Vice President Moses Blah, but he and Speaker of the House Nyudueh Morkonmana are jostling for power.

Copyright 2003 CNN. All rights reserved. This material may not be published, broadcast, rewritten, or redistributed. Associated Press contributed to this report.



Peacekeepers expect Taylor to leave power

Rebel leader: Opposition does not trust Liberian president

MONROVIA, Liberia (CNN) -- A **West** African **organization** that sent an initial contingent of **peacekeepers** to Liberia expects President Charles Taylor to abandon power once the force is deployed, a **spokeswoman** for the regional bloc said Monday.

The first group of 300 Nigerian **peacekeepers** from **the Economic Community** of **West** African States (ECOWAS) was greeted upon its arrival in Liberia earlier in the day by hundreds of cheering, war-weary

A Nigerian peacekeeper is carried by a cheering crowd of Liberians

But, Conneh said, the rebels "do not trust Charles Taylor."

"He has said he is leaving on several occasions and he is not leaving," Conneh said. "But this time we are very happy that an **international** force is on the ground.

"You will not have to worry about the rebels if he does not leave," he said. "The **international** force will deal with it."

Conneh also **said LURD** would disarm once a transitional leader replaced Taylor.

Taylor, elected president six years ago after he emerged as the strongest warlord in a devastating, **nine-year-long** civil war, initially agreed to step down in June when he was indicted on the war crimes charges.

He told reporters Saturday after a two-hour meeting with a **West African** delegation that he would hand over power to his successor at 11:59 a.m. August 11 and that the new president would be sworn in at midday.

He refused to answer questions about whether he would honor an earlier pledge to accept exile in Nigeria.

Liberia's National Assembly will meet Thursday to pick a new president, Taylor said. Under **Liberia's** constitution, that role should fall to Vice President Moses Blah, but he and Speaker of the **House Nyudueh** Morkonmana are jostling for power.

Copyright 2003 CNN. All rights reserved. This material may not be published, broadcast, rewritten, or redistributed. Associated Press contributed to this report.

Fig. 12. The actual layout of a web page. **Fig. 13.** The same web page "blindlighted".

Thus, a new proposal to perform some NLP tasks such as language identification, document clustering, keyphrase extraction and text summarization has been presented with particular emphasis in the last two tasks. The described techniques are based primarily on biological phenomena, and they show that naïve keyphrase extraction and summarization algorithms perform feasibly without human guidance and no more than free text from a single document.

Table 2. Results for the preliminary test on automatic keyphrase extraction.

Journal	Mean author's keywords in contents	Mean matches	Max. matches	Min. matches
B&BSPA	78.5%	40.4%	58.5%	33.0%
Psycoloquy	55.3%	36.8%	66.7%	00.0%
JIAHR	87.5%	50.2%	65.0%	31.4%

[7] One article from *Psycoloquy* provided eight keyphrases and only one (12.5%) was mentioned in the article. There were many lists of keywords with less than 20% of them in the contents.

References

1. Dunning, T.: Statistical identification of language. Computing Research Laboratory, New Mexico State University (1994)
2. Sibun, P., Reynar, J.C.: Language Identification: Examining the Issues. Proc. of 5th Symposium on Document Analysis and Information Retrieval. Las Vegas, USA (1996)
3. Cavnar, W.B., Trenkle J.M.: N-Gram-Based Text Categorization. Proc. of 3rd Annual Symposium on Document Analysis and Information Retrieval, Las Vegas, USA (1994)
4. Slonim, N., Tishby, N.: Document clustering using word clusters via the information bottleneck method. Proc. of 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Athens, Greece (2000) 208-215
5. Hulth, A.: Improved Automatic Keyword Extraction Given More Linguistic Knowledge. Proc. of the 2003 Conference on Empirical Methods in NLP. Sapporo, Japan (2003)
6. Matsuo, Y., Ishizuka, M.: Keyword Extraction from a Single Document using Word Co-occurrence Statistical Information. Proc. of the Sixteenth International FLAIRS Conference. St. Augustine, FL, USA (2003) 392-396
7. Hovy, E., Lin, C.: Automated text summarization in SUMMARIST. In: I. Mani and M. Maybury (eds.), Advances in Automated Text Summarization. MIT Press (1999)
8. Srinivasan, R., Rose, G.D.: Ab initio prediction of protein structure using LINUS. Proteins: Structure, Function, and Genetics, Wiley-Liss, 47(4) (2002) 489-495
9. Larson, S.M. et al: Folding@Home and Genome@Home: Using distributed computing to tackle previously intractable problems in computational biology. In: Richard Grant (ed.), Computational Genomics. Horizon Press (2003)
10. Ferreira da Silva, J., Pereira Lopes, G.: Extracting Multiword Terms from Document Collections. Proc. of VExTAL, Venice, Italy (1999)

Appendix: Algorithms

Algorithm precalculateMatrix *(ngramSequence)*
Input: the list of weighted n-grams *ngramSequence*

```
1.  totalNgramsWeight ← 0
2.  for each n-gram ngram in ngramSequence do
3.      totalNgramsWeight ← totalNgramsWeight + ngram.weight
4.      firstSegments ← getFirstSegments(ngram)
5.      secondSegments ← getSecondSegments(ngram)
6.      from i ← 0 to size of firstSegments do
7.          first ← firstSegments(i)
8.          sec ← secondSegments(i)
9.          precalcMatrix(first)(sec) ← ngram.weight
10.         precalcMatrix(first)(ALL) ← precalcMatrix(first)(ALL) + ngram.weight
11.         precalcMatrix(ALL)(sec) ← precalcMatrix(ALL)(sec) + ngram.weight
12.     loop
13. loop
14. totalSegmentsWeight ← 0
15. for each value weight in precalcMatrix(ALL) do
16.     totalSegmentsWeight ← totalSegmentsWeight + weight
17. loop
```

***Algorithm SI_f** (ngram)*
Input: the object *ngram* storing an n-gram plus its weight (no. of appearances in document)

```
1.  $pw_1w_n$ ← ngram.weight / totalNgramsWeight     // totalNgramsWeight is global
2.  firstSegments ← **getFirstSegments**(ngram)
3.  secondSegments ← **getSecondSegments**(ngram)
4.  avp ← 0
5.  **from** i ← 0 **to** size of firstSegments **do**
6.      $w_1w_i$ ← firstSegments(i)
7.      $w_iw_n$ ← secondSegments(i)
8.      $pw_1w_i$ ← precalcMatrix($w_1w_i$)(ALL) / totalSegmentsWeight
9.      $pw_iw_n$ ← precalcMatrix(ALL)($w_iw_n$) / totalSegmentsWeight
10.     avp ← avp + $pw_1w_i$ * $pw_iw_n$
11. **loop**
12. avp ← avp / (sizeNgram – 1)     // sizeNgram is global
13. **return** $\log_{10}$ ($pw_1w_n$ / avp)  // Fair Specific Mutual Information value for ngram
```

***Algorithm ribosomalTranslation** (text, tRNA, sizeNgram)*
Input: the contents, *text*, from a document, a hash table *tRNA* which associates a significance value to an n-gram, the size of the n-grams *sizeNgram*

```
1.  i ← 0
2.  candidateKey ← λ// empty string
3.  oldCandidateSignificance ← 0
4.  **while** i < (length of text – sizeNgram) **do**
5.      chunk ← substring (text, i, sizeNgram)
6.      candidateKey ← **merge**(candidateKey, chunk)        // Combines two strings
7.      acumSignificance ← acumSignificance + tRNA(chunk)
8.      newCandidateSignificance ← acumSignificance / length of candidateKey
9.      **if** newCandidateSignificance > oldCandidateSignificance
10.         oldCandidateSignificance ← newCandidateSignificance
11.         i ← i + 1
12.     **else**
13.         candidateKey ← candidateKey – chunk
14.         Call **addNewChunk** (candidateKey, oldCandidateSignificance)
15.         candidateKey ← λ
16.         oldCandidateSignificance ← 0
17.         i ← **undo**(i, chunk)    // Undo the last action to recover i value
18.     **end if**
19. **loop**
```

***Algorithm proteinFolding**(textChunks)*
Input: the hash table *textChunks* that associates a significance weight to each text chunk

```
1.  newKeys ← **selfAttract**(textChunks)      // Obtains a list of chunks with new weights
2.  threshold ← mean of newKeys weights + π/2 * typical deviation of newKeys weights
3.  window ← **round**($\log_2$(size of newKeys))
4.  newKeys ← **mergeKeys**(newKeys, threshold, window)      // Merges similar keys
5.  threshold ← mean of newKeys weights + π/2 * typical deviation of newKeys weights
6.  newKeys ← **sliceKeys** (newKeys, threshold)
7.  **return mergeKeys** (newKeys, 0, size of newKeys)  // Subset of weight keyphrases
```

***Algorithm selfAttract**(textChunks)*

Input: the hash table *textChunks* that associates a significance weight to each text chunk

```
1.  // Each chunk is assigned  a “probability” of attraction based on its original ranking
2.  max ← $(\pi/2)^{size\ of\ textChunkss} - 1$
3.  from i ← 0 to size of textChunks do
4.      tmpPweights(i) ← $(\pi/2 * (\pi/2)^i)$ / max
5.  loop
6.  for each keyphrase key in textChunks do
7.      pWeights(key) ← tmpPweights(ranking(key, textChunks))
8.  loop
9.  // Each chunk is assigned a “probability” of attraction based on its contacts weights
10. maxContact ← 0
11. for each keyphrase $key_1$ in textChunks do
12.     contacts ← 0
13.     for each keyphrase $key_2$ in textChunks do
14.         if partialMatch ($key_1$, $key_2$)
15.             contacts ← contacts + textChunks($key_2$) / textChunks/($key_1$)
16.         end if
17.     loop
18.     if contacts > maxContact
19.         maxContact ← contacts
20.     end if
21.     pContacts($key_1$) ← contacts
22. loop
23. for each keyphrase key in textChunks do
24.     pContacts(key) ← ( maxContact – pContacts(key)) / maxContact
25. loop
26. // The final attraction “probability” is pWeights * pContacts
27. for each keyphrase key in textChunks do
28.     pAttraction(key) ← pWeights(key) * pContacts(key)
29. loop
30. // New weights after attraction reinforcements are compute
31. for each keyphrase $key_1$ in textChunks do
32.     $w_1$ ← textChunks($key_1$)
33.     for each keyphrase $key_2$ in textChunks do
34.         if partialMatch ($key_1$, $key_2$)
35.             $w_1$ ← $w_1$ + pAttraction($key_1$) * pAttraction($key_2$) * textChunks($key_2$)
36.         end if
37.     loop
38.     newWeights($key_1$) ← $w_1$
39. loop
40. return newWeights  // Input keyphrases with new weights after mutual reinforcement
```

Algorithm enrichKeyphrases *(keyphrases, document, maxIters)*
Input: the hash table k*eyphrases* with weighted keyphrases, *document*, the contents of the document and the maximum number of iterations *maxIters*

```
1.  iters ← 0
2.  while new keyphrases are found and iters < maxIters do
3.      for each keyphrase key in keyphrases do
4.          leftSiders ← getLeftSideWords(key, document)
5.          rightSiders ← getRightSideWords(key, document)
6.          for each left side word l in leftSiders do
7.              keyPairs(l)(key) ← keyPairs(l)(key) + 1
8.          loop
9.          for each right side word r in rightSiders do
10.             keyPairs(key)(r) ← keyPairs(key)(r) + 1
11.         loop
12.         drop any keyPairs(a)(b) with a value of 1
13.         keyphrases ← merge(keyphrases, keypairs)
14.         iters ← iters + 1
15.     loop
16. loop
17. return keyphrases    // The hash table updated with new rich keyphrases
```

Algorithm blindLight *(passages, richKeyphrases, K)*
Input: the list of passages from the text *passages*, the hash table *richKeyphrases* with weighted refined keyphrases and the *K* factor to modify the summary threshold

```
1.  for each passage p in passages do
2.      acumWeight ← 0
3.      matches ← 0
4.      for each rich keyphrase key in richKeyphrases do
5.          if key is in p
6.              acumWeight ← acumWeight + richKeyphrases(key)
7.              matches ← matches + 1
8.          end if
9.      loop
10.     highlightWeights(p) ← matches * acumWeight / length of p
11. loop
12. threshold ← K * mean of non-zero highlightWeights
13. for each passage p in passages do
14.     if highlightWeights(p) > threshold
15.         Call highlightHTML(p)
16.         Call addPassageToSummary(p)
17.     end if
18. loop
```