International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 8 (2016) pp. 235-246
© MIR Labs,www.mirlabs.net/ijcisim/index.html

RAID: Robust Algorithm for stemmIng text Document

Kabil BOUKHARI[1] and Mohamed Nazih OMRI[2]

[1]MARS Unit of Research,
Department of computer sciences
Faculty of sciences of Monastir,
University of Monastir, 5000, Tunisia
kabil.boukhari@gmail.com

[2]MARS Unit of Research,
Department of computer sciences
Faculty of sciences of Monastir,
University of Monastir, 5000, Tunisia
Mohamednazih.omri@fsm.rnu.tn

Abstract: **In this work, we propose a robust algorithm for automatic indexing unstructured Document. It can detect the most relevant words in an unstructured document. This algorithm is based on two main modules: the first module ensures the processing of compound words and the second allows the detection of the endings of the words that have not been taken into consideration by the approaches presented in literature. The proposed algorithm allows the detection and removal of suffixes and enriches the basis of suffixes by eliminating the suffixes of compound words. We have experienced our algorithm on two bases of words: a standard collection of terms and a medical corpus. The results show the remarkable effectiveness of our algorithm compared to others presented in related works.**

Keywords: Robust algorithm, Stemming, Documents indexing, Information retrieval.

I. Introduction

In recent years, electronic documents have increased both in the Internet and in corporate intranets. Finding the information needed here in thus proves a task increasingly difficult. Often, users are discouraged by the slowness and inefficiency of traditional search engines available. Automatic indexing of documents makes it easy and solves much of the problem.
Automatic indexing is defined as a document representation of the analysis results of natural language or standardized language of a document [19][20][21]. Another more classic definition and consonant definition suggests that automatic indexing is the identification and location of relevant sequences or major themes in a document by analyzing its content [22][23][24].However, other works [25] [26] have shown the existence of irrelevant concepts for texts.
The indexing phase is subsequently classified using indexes, the document from a set of documents in a given collection and retrieving the context of this index within the document itself . This type of indexing is to optimize access to data in large databases[1].
In this context, the research is centered on the extraction of key terms used in the documents to facilitate access and navigation in web pages and to find the electronic information. These keywords are used in the process of information search to get relevant answers to questions[2]. The questions that arise are of the form: Can we find this document? How well the documents are relevant? Do they meet user needs?
To answer these questions, the system must take the user input in the form of key terms and linking them to information contained in the documents.
The recovery technique paves the way for a possible inquiry about the fact if any given document and a given query share a particular keyword or not. The obvious answer is simply tested for absolute equality between the keyword and all terms in the document. It is only after the confirmation of an existing similarity is found that automatic indexing retrieves it.
However, the terms key can have many morphological variants that share the same semantics and can be beneficial for the information retrieval system to consider these equivalent terms. To recognize these variations, as in [3][33]the system needed terms in a natural form in order to treat them. A form of natural language processing, which may be opted for to carry out this task, can be an algorithm that transforms an end to its morphological root via the removal of prefixes and suffixes[4]. Here we can talk about stemming.
Then the purpose of an information retrieval system (IRS) [30][31][32] is to find the most relevant documents that correspond to the user's queries given the large number of

documents on the Internet. So, to improve the performance of an IRS, it is essential to develop a new stemmer for a more relevant and accurate indexing system.
The techniques used[5] are generally based on a list of affixes (suffixes, prefixes, postfix, antefixes) of the language in question and on a set of stemming/de-suffixation rules constructed already, that allow the finding of the stem of a word.
Several algorithms have been proposed for research lexical root for the English language. The main algorithms developed in this senseare the LOVINS, Paice, Husk, PORTER, EDA and Krovetz algorithm's.
Part of the focus of this work is the study of two standard algorithms, namely LOVINS algorithm and that of PORTER. We present the definition and the principle of each of the five algorithms.
As for stemming algorithms, there is no perfect algorithm that meets user needs for different corpus. Meanwhile, this algorithm allows the indexing process of non–structured documents.
The focal blemish of de-suffixation algorithms that are developed so far is their lack of producing one hundred percent reliable results (lack of precision): same context words do not have the same stems. We noticed that, on the one hand, in the stemming process, there is no phase of treatment for compound shapes, and on the other hand, several suffixes are ignored and are not, therefore, treated, which is the case of the LOVINS algorithm.
This paper is divided into fourparts. After the introduction, the second presents the stemming phase of the unstructered documents. The third part devoted to the presentation of the best-known stemming algorithms in literature, this section will be concluded by a comparative study of different algorithms and their limits. Part four presents the proposed algorithm, inspired from the SAID approach[27], for stemming words of a text. The fifth paragraph presents the experimental data, the results obtained and provides a detailed analysis of these results. We finish this work by a conclusion and we give the perspectives of the future work.

II. Stemming

The label stemming or de-suffixation is given to the process that aims to transform the inflections in their radical or stem. It seeks to bring together the different variants, inflectional and derivationnel, of a word around a stem.
The root of a word corresponds to its remaining part, namely its stem, after the removal of its affixes (prefix and/or suffix). It is also sometimes known as the stem of a word. Unlike the lemma that corresponds to a real word in the language, the root or stem is generally not a real word. For example, the word "relation" has "rel" as a stem which does not correspond to a real word but in the example of "greatest" the radical or stem is "great".
The stemming is an important stage for the indexing process [6][7]. De-suffixation algorithms have been developed to effectively treat a given problems (slow response time, numerous documents, lack of precision). These algorithms are designed to identify the roots of the words through a set of rules and conditions.
Stemming operation consists of removing inflectional and derivational suffixes to reduce the various forms of a word at their root. This root must be included in a morphological sense: two words might share or have the same morphological root and completely different meanings.
The techniques used are generally based on a list of affixes of the language at hand as well as a set of de-suffixation rules priory built that allow the finding of a stem of a given word.
Search engines use stemming algorithms to improve information retrieval[8]. The keywords of a query or document which are represented by their roots rather than by the original words. As in [9] several variations of a term can thus be grouped into a single representative form, which reduces the size of the dictionary, and then the number of distinct words needed to represent a set of documents. A dictionary of reduced size saves both space and execution time.
There are two main families of stemmers: the algorithmic stemmers and dictionary-based stemmers: algorithmic stemmer is often faster and can extract the roots of unfamiliar words (in a sense, all found words are unfamiliar to it). However, it will have a higher error rate, grouping sets of words that should not be together (over-Stemming). A dictionary based stemmer where the number of error on known words is almost zero. It is also slower and requires the removal of suffixes before looking for the corresponding root in the dictionary.
The de-suffixation algorithm functions on different steps through which the words to process successively pass, according to the rules, when the parser recognizes a suffix from the list, it removes or transforms it. Here the longest suffix which determines the rule to be applied. Each algorithm has its own steps and its different rules.

III. Related works

Different Stemming algorithms [10] have been proposed in literature.
The first algorithm that we treat is Husk and Paice's algorithm [11][12] and belongs to the family of algorithmic stemmers. It is a simple iterative Stemmer, it removes the suffixs from a word in an indefinite number of steps.
This is algorithm based on a set of rules to extract roots with more stores outside the rules of the code. On each iteration, it tries to find an applicable rule by the last character of the word.
This approach consists of a set of functions that will use the root extraction rules applicable to the input word and check the acceptability of the proposed root, and the set of rewrite rules. The main function takes as parameters the word that we want to extract, the root and the code of the language.
The second algorithm treated is EDA anddeveloped by Didier Nakache et al. [13]. It is used to de-suffix medical corpus in french. It works in two main phases: a phase of

preparation and a phase of harmonization of the form followed by a phase of treatment.

The first phase serve to prepare the word to be stemmed by applying some modifications:

1. transformation in lowercase,
2. separation of ligated characters and hyphens,
3. removal of diacritics.
4. removal of double letters.
5. replace some letters by others according to rule.

This first phase allows to cleans the term and puts it into a ‘standard’ form, The removal of accents used to group many terms that were considered, in advance, as different. And the last three preparation rules allow only to correct typing errors and errors induced by change case.
The second phase serve to execute a set of rules. It is important to follow these rules in order, until the resulting Word contains 5 characters or otherwise, up to the last rule. Each rule applies to the result obtained by the previous rule.

The third algorithm is Krovetz [14], which is considered as a "light stemmer" because it uses inflectional language morphology. It's a low strength algorithm and complicated due to the processes involved in linguistic morphology and its inflectional nature.
The area of morphology can be broken down into two subclasses, inflectional and derivational. Inflectional morphology describes predictable changes a word undergoes as a result of syntax (the plural and possessive form, past tense…). These changes have no effect on a word’s ‘part-of-speech’ (a noun still remains a noun after pluralizations). In contrast, changes of derivational morphology may affect /may not affect a word’s meaning. Although English is a relatively weak morphological language compared to other languages have stronger morphology where thousands of variants may exist for a given word. In such a case the retrieval performance of an Information Retrieval system [35] would be severely be impacted by a failure to deal with such variations.

“Krovetz”removes effectively and specifically suffixes:

- The conversion of a plural to its singular form (e.g: '-ies', '-es', 's').
- The conversion of past to present time (e.g: '-ed').
- The elimination of '-ing', then through a verification process in a dictionary for any recoding and returns the word stem.

This Stemmer is often used in conjunction with any other "Stemmers" taking advantage of the accuracy of removal of suffixes by this algorithm. Then, it adds the compression of an another "Stemmer" like the Paice/Husk algorithm or PORTER.

The PORTER algorithm [15][16] is the fourth one we have studied. It is the most famous stemming algorithms which can eliminate the affixes of words to get a canonical form of the latter. This algorithm is used for the English language, but its effectiveness is very limited when it comes to treating the French language, for example, where the inflections are more important and more various.
The PORTER algorithm consists of fifty Stemming rules classified into seven phases (treatment of plurals and verbs in the third person singular, the past and the progressive treatment,...). The words to be analyzed pass through all the stages and, in the case where several rules may be applied to them, this is always the longest suffix is chosen. The de-suffixation is accompanied, in the same stage of recoding rules.
Thus, for example, "troubling" will become "troubl' by the removal of the progressive marker suffix ‘– ing’ and will be then transformed into "trouble" by application of the rule "bl" becomes "ble". This algorithm includes also five context rules, which indicate the conditions in which a suffix should be omitted. The ending ‘-ing’, for example, will not be removed unless the radical has at least a vowel. In this way, "troubling" will become 'troubl', while "sing" remains "sing".
The use of the term in an inflected language, it gives inflections. Inflexion is a morphological modification of a term to mark the grammatical position, the tense of conjugation... For example, the verb “play” inflectes in “played” when it is placed in a sentence in the past with all persons, in the singular and the plural. The word “baby” inflects in “babies” in the plural.
PORTER has been developed to permit the application of the rules defined in a particular syntax on inflected words. The application of rules allows for morphological transformations in order to obtain a standardized version from a flexed release.
A new version of PORTER [17] has been developed to improve the original stemming algorithm. The objective of the Stemming is to find the common canonical form for inflected words.

The last algorithm studied is the LOVINS algorithm [18], which has 294 suffixes, 29 conditions and 35 transformation rules and where each suffix is associated with one of the conditions. In the first step, if the longer ending found satisfies its condition associated therewith, the suffix will be eliminated. In the second step, the 35 rules are applied to transform the suffix. The second step is performed whether thesuffix is removed in the first step or not.
For example, “nationally” has the suffix “ationally”, with associated condition, B, ‘minimum length of stem = 3’. Since Remove 'ationally' leaving a stem with a single letterthen this condition is rejected. But it also has ending “ionally” with associated condition A. Condition A is ‘no restriction on stem length’, so “ionally” is removed, leaving “nat”.
The transformation rules handle features like letter undoubling, irregular plurals and English morphological oddities ultimately caused by the behaviour of Latin verbs of the second conjugation.
The stems are grouped according to the length of 11 characters up to 1, each termination is followed by its

condition code. There is an implicit assumption in each condition, is the length of the root must be at least equal to 2 characters.

An other version of Lovins Stemmer : The Dawson stemmer [34]. This approach is similar to Lovins as it is a single-pass context-sensitive suffix removal stemmer. The main aim of the stemmer was to take the original algorithm proposed by Lovins and try to refine the rule sets to correct any basic errors that exist.
The first step is to include all plurals and combinations of the simple suffixs, this increased the size of the suffixs list. The second phase is to employ what Dawson called the completion principle in which any suffix contained within the ending list is completed by including all variants, flexions and combinations in the ending list. This increased the ending list once more , although no record of this list is available.

A similarity with the Lovins stemmer is that every suffix contained within the list is associated with a number that is used as an index to search an list of exceptions that enforce certain conditions upon the removal of the associated ending. These conditions are similar to the Lovins algorithm in that they may enforce either a minimum length of the remaining stem or that the suffix can only be removed/shall not be removed when set letters are present in the rest of the stem.
The major difference between the Dawson and Lovins approach is the technique used to solve the problem of spelling exceptions. Lovins uses the technique known as recoding. This process is seen as part of the main algorithm and performs a number of transformations based on the letters within the stem. In contrast Dawson uses partial matching which, as described above, try to match stems that are equal within certain limits. This process is not seen as part of the stemming algorithm and therefore must be implemented within the information retrieval system. Dawson warns that without this additional processing many errors would be produced by this stemmer.

Although PORTER and LOVINS are known by their power they still face many problems .
For the PORTER stemmer the main poblem is when many words derived from the same root do not have the same stem. This is due essentially to the fact that PORTER stemmer ignores many cases and disregards many exceptions. In addition, PORTER stemmer does not treat irregular verbs, Irregular plural nouns are not handled by the stemmer: words ending with ‘men’ are the plural of words ending with ‘man’. Many exceptions are not controlled: verb conjugation, possessive nouns, irregular comparative and superlative forms (e.g. good, better, best), etc. Moreover, many suffixes are not handled by this alogorithm. This would decrease the stemming quality, since related words are stemmed to different forms.
The main limitations detected in LOVINS's algorithm can be summarized in the following points: disregard of compound words (childhood, relationship, chairman ...), several missing suffixes for different lengths, the elimination of doubling of characters (LOVINS is not taken into account 10 letters of the alphabet) and insufficient processing rules. The LOVINS algorithm, for example, ignores the words of compound shapes (Compound Words), the suffixes of these words can be classified by length, and a set of words in same context must have the same stem.
In an information retrieval context, such cases reduce the performance since some useful documents will not be indexed with some terms, This would decrease the efficiency of diverse indexing systems applying those stemmers.

IV. RAID: Robust Algorithm for stemmIng text Document

In order to improve performance stemmers, we studied the English morphology, and used its characteristics for building a new stemmer capable to improve information retrieval systems and indexing systems.
The study conducted in the previous paragraph on stemming algorithms has enabled us to identify the advantages and disadvantages of each of these algorithms. We focused in particular the limits presented by the two best known and most used algorithms namely PORTER and LOVINS' algorithm. Errors made by these stemmers may affect the information retrieval performance.
Observing the main stemming algorithms in the literature, we found that they are based on the best-known suffixes and the most used ones in the morphology of the English language. Some cases have not been investigated which generate the non consideration of a large number of suffixes (approximately 140 suffixes), and thus transformation rules have not been set up and were not considered.

A. *Stemmer algorithm «RAID»*

```
Algorithm : RAID
Intializations
    F ← File of words
    CW← Compound word
    LS← List of suffixs
    S←Suffix of a word
    T← Transformation
    ST← Stem
    W← Word from the corpus
    R← Rule
    DC← List of double characters

Functions
    Remove_suffix(x,y) : Remove x from the word y
    Apply_rule(x,y) : Apply the rule x to the term y
    WS(x,y) :Extract the suffix x from the word y

Inputs:
    Set of the words

Outputs:
    Set of the stemmed words
```

Treatement :

```
Begin
while (∃ W in F ) do
    ST←W
  /* Step 1 */
        if (∃ST ∈ CW) then
            ST←Remove_suffix(CW,ST)
        Endif
  / * Step 2 Determine the location in the list of endings * /

  / * Step 3 Find a correspondence between the word and
          one of the suffix endings in the list * /
        if (WS(S,ST) ∈ LS) then
            Suffix_found←True
        Endif

  / * Step 4 * /
        if (Suffix_found) then
            ST←Remove_suffix(S,ST)
        Endif

  / * Step 5 remove doubling if exists / *
        if (WS(S,ST) ∈ DC) then
            ST←Remove_suffix(DC,ST)
        Endif

  / * Step 6 * /
        if (∃T) then
            ST←Apply_rule(R,ST)
        Endif

  Return (ST)
End while
EndRAID
```

Algorithm 1. RAID : Robust Algorithm for stemmIng text Document

For our approach, we proposed to add a phase to treat compound words and other phase to enrich the basis of existing suffixes with a new base of suffixes (over 100 suffixes for single words and approximately 40 suffixes for compound words) identified in the conducted study. We have defined a set of transformation rules to the set of words for which we have detected new suffixes.

After the phase of treatment of derivational morphology, a new base has been composed to treat compound words, it maps complex terms (term compound of more than one word) to a single root from which they were derived (e.g. transform the term ‘businessman’ to ‘business’).
For the elimination of doubling, these algorithms presented only a few characters that can be doubled but in reality, in the English language, there are several letters that can mark the doubling at the end of a Word, for this we have added the missing characters that can be doubled.
A new rules base, which enriches the old morphological basis of the English language, represents our third contribution. We discovered another suffix with a length equal to 12 that we have considered in the process of our algorithm development.

Our algorithmis articulated around four stages:

- ✓ Checking word if composed or not, whether elimination of the composition..
- ✓ Searches the list of suffixes, correspondence with the ending of the word to be stemmed, and application of the correct rule.
- ✓ Elimination of doubling there.
- ✓ Application of one of the transformation rules and returns the stemmed word.

B. *Treatment of compound words*

In this stage, we identified the basic suffixes for compound words of this language. We have built a new suffixes base which enrich the former base used by most of algorithms. This is the first contribution of our approach to stem compound words.
These algorithms, for example, ignore the words of compound shapes (Compound Nouns), the suffixes of these words can be classified by length, and a set of words in same context must have the same stem.

Example :

Context	**Original word**	**Result**
	relate	rel
	relates	rel
	relating	rel
	relation	rel
Context 1	relational	rel
	relations	rel
	relationship	rel
	relationships	rel
	chair	chair
Context 2	chairs	chair
	chairman	chair

*Table 1.*Example of treatment of compound words by the algorithm RAID

According to the previous example, we note that the words such as "relationship", "relationships" and "chairman" are not stemmed by the LOVINS algorithm. We then took into account this limit in order to get around in the proposed algorithm RAID.

C. *Treatment of the ignored suffixes*

In this part, we proposed to create a new basis of ignored and existing suffixes. After experimentation, we found that the LOVINS algorithm has not processed several suffixes, like ‘ativistic’, ‘itivness’ ‘iations’,‘itively’, ‘ements’, ‘ition’, ‘ele’, ‘er’… ,and according to the provided result some affixes are not indicated in this algorithm.

Example:

great → great

greater	→	greater
greatest	→	great
greatly	→	great

For the word "greater", for example, the LOVINS algorithm ignores the suffix "er". We then have considered this second limit by applying the correct rule. In each suffixes class, we identify the most used and most suitable affixes/endings to our corpus.

D. Transformation rules associated to conditions

In this section, we present 29 conditions, called A to Z, AA, BB and CC and each condition is associated with a rule.
For the transformation rules, LOVINS suggested a base of 35 rules that aim to transform a suffix in another suffix. We found that these rules are limited in number because we have identified other rules in English literature. These latest transformation rules have been used in our algorithm RAID.
For the elimination of the doubling, the LOVINS and Porter algorithms, for example, offers 10 characters that can be doubled, in fact there are more than 10 characters which are ignored by these algorithms and which can be doubled to do this. Hence, we added 10 additional characters.

Example:

For the word 'fuzzy', which has the suffix 'y', we apply the rule 'B': eliminate the suffix if the length of the stem is at least 3 characters then 'fuzzy'$\rightarrow$ 'fuzz'! No rule of transformation to eliminate doubling 'zz'.

For a word that has a suffix of length 1, we apply the rule 'w' on the word 'cliffs'. We get 'cliff'. LOVINS algorithm does not allow the elimination of doubling for the letter 'f'.

For this we added some rules for the elimination of doubling
After treatment of the corpus and the provided results, we were able to extract some other processing rules.

Example:

For the two words in same context 'flagstaff and flagstaves', after the stemming phase, these algorithms gives 'flagstaff and flagstav'then according to this result the new transformation rule will be:

Stav $\rightarrow$Staf

Same principle to other rules that we have identified.

V. Experimental results and discussion

This part describes the various tests carried out and the different results obtained. A first exprementations for a standard database test of English language and a second for a medical corpus.

A. Experimentation for a standard database test

The development of stemmers aimed to improve indexing systems performance by transforming terms semantically equivalent to a single stem. This infers that an efficacious stemmer should conflate only pairs of words which are in the same context. The problem is how the algorithm will distinguish between two words when they are semantically equivalent. Paice [11] proposed a solution to provide an input to the program in the form of grouped file. The data file contains a list of words sorted alphabetically and terms that are semantically equivalent. A set of words in the same context forms a group.
If a group of stems contain more than a single root, we can talk about error 'understemming'. Various tests of performance of our algorithm RAID were conducted and compared to PORTER and LOVINS algorithms.
To implement our algorithm, we used DEV c++.We used a corpus containing approximately 10000 words (Paice solution),in the English-language database.
For a more detailed analysis, we analyse the structure of the word list. We find that an important number of words are related to the derivational morphology. PORTER and LOVINS ignore many derivational suffixes. This problem was resolved our proposed approach. RAID stemmer handles very well derivations and inflections.
Table 2 below shows error rate respectively registered by PORTER, LOVINS and RAID:

Algorithm	**Total number of words**	**Number of irrelevant terms**	**Error rate**
PORTER	9717	5600	0.5763
LOVINS	9717	4909	0.5051
RAID	9717	4210	0.4332

*Table 2.*Error rate Registered by the algorithms Porter, Lovins and RAID

According to the results in table 2, we note that for three algorithms, error rate is important. However, it is clear that the error rate (over stemming and indestemming) of our algorithm is significantly lower than that recorded by the algorithms of LOVINS and PORTER. We notice that the difference between the approaches is increasingly important when the number of terms in the basic tests increases.

Algorithm	**Total number of relevant terms**	**Relevant terms correctly attributed to the Ci[1] classes**	**Irrelevant terms attributed to Ci classes**
PORTER	6678	4117	2561
LOVINS	7323	4808	2515
RAID	7756	5507	2249

Table 3. Number of terms extracted by the algorithms PORTER, Lovins and RAID

From the table as mentioned above, We can make it right that our algorithm can cover 7756 terms which are irrelevant terms in 2249 for all classes of the dataset. This reduction of noise compared to LOVINS and PORTER

[1]Ci Aclass corresponds to a set of words in the same context

algorithms is due to the addition of a number of suffixes and to the integration of compound words.

To evaluate the performance of RAID, we used three standard performance measures namely precision, recall and F-mesaure:

Precision: is the ratio between the number of relevant terms correctly attributed to the classes Ci (NRTC) and the total number of relevant terms in the corpus (NTTC) (1).

Recall: is the ratio between the number of relevant terms correctly attributed to the classes Ci (NRTC) and the total number of terms (NTT) (2).

F-measure: is the harmonic average that combines precision and recall (3).

$$\textbf{Precision}= NRTC/NTTC \qquad (1)$$

$$\textbf{Recall}= NRTC/NTT \qquad (2)$$

$$\textbf{F-measure}= \frac{2*(Precision*Recall)}{(Precision+Recall)} \qquad (3)$$

We have varied the size of the set of terms used from 500 words to 10,000 words in order to study the behavior of each of the algorithms, in particularly RAID algorithm.

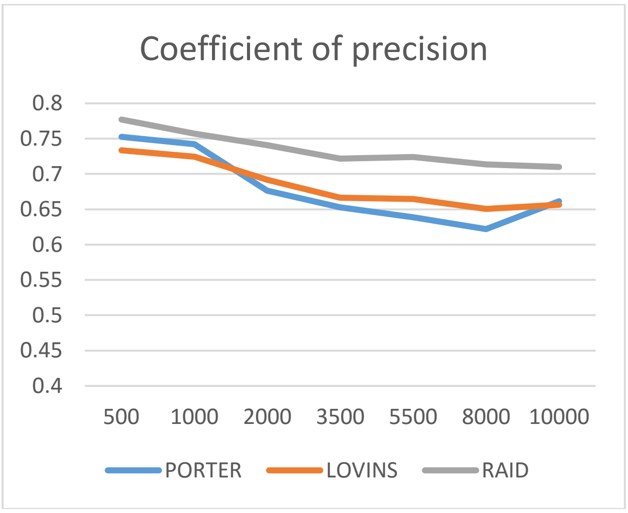


Figure 1. Precision rate of PORTER, LOVINS and RAID algorithms

We can notice that the RAID algorithm, has an important advantage for accuracy by reducing noise and the number of irrelevant terms.

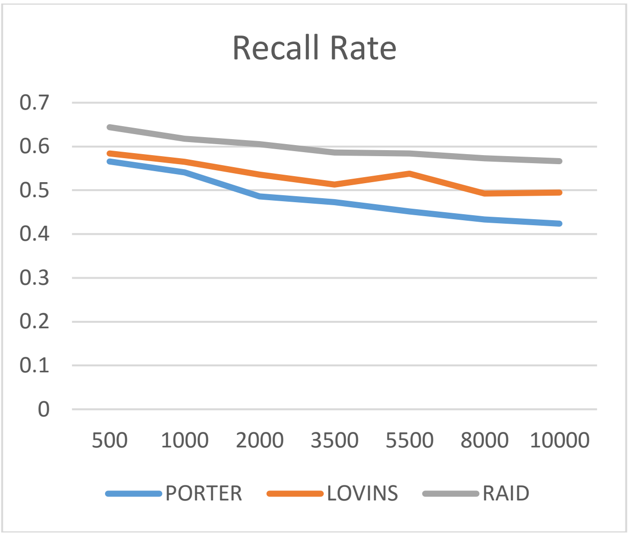


Figure 2. Recall rate of PORTER, LOVINS and RAID algorithms

The Recall provided by RAID is important also, with a remarkable superiority over the LOVINS and PORTER algorithms. Indeed, it reduces the silence factor to meet the need for information and gives the yearned results.

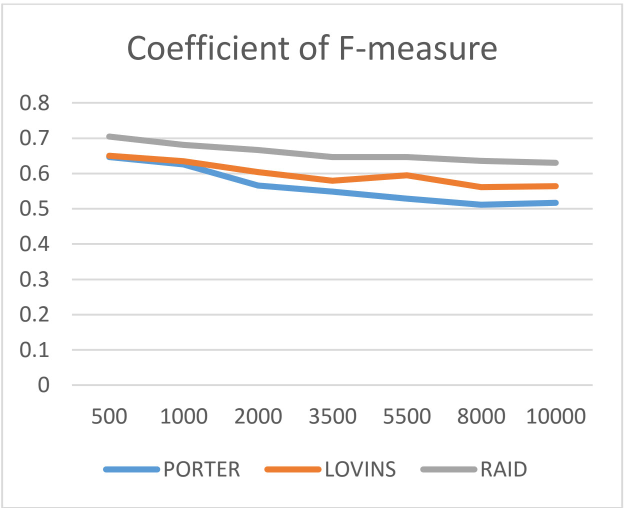


Figure 3. F-mesasure rate of PORTER, LOVINS and RAID algorithms

For F-measure rate, we note that RAID is more accurate than LOVINS and PORTER. This interesting result is due to the inclusion of the various suffixes, which allow to improve RAID and to provide more relevant terms, generally ignored by other algorithms.

B. Experimentation for a medical corpus

To improve the performance of an indexing system [28], in the preprocessing phase, a stemming algorithm is almost necessary. The first advantage of this pretreatment is to reduce the size of the indexes database or the dictionary size. And on the other hand with a stemmer, several

morphological variants of a term can be grouped into a single representative form. This creates a better indexing.
In this experimental section, we used a medical corpus contains 10000 from MESH thesaurus [29].
MeSH (Medical Subject Headings) is the National Library of Medicine's controlled vocabulary thesaurus used for indexing articles for PubMed.
The word list used was downloaded from the official website National library of Medicine. The terms of the list are grouped by context, a set of temrs semantically equivalent are grouped together.
In this work, we make tests initially with PORTER and LOVINS stemmers and RAID stemmer in order to evaluate our approach. The results of these tests are presented in the following table .

Algorithm	Total number of words	Number of irrelevant terms	Error rate
PORTER	10000	3797	0.37
LOVINS	10000	3245	0.32
RAID	10000	2904	0.29

Table 4. Number of terms extracted correctly by the algorithms Porter, LOVINS and RAID for the medical corpus

According to the results obtained above, we note that although the error rate of RAID stemmer is low compared to the PORTER and LOVINS algorithms.

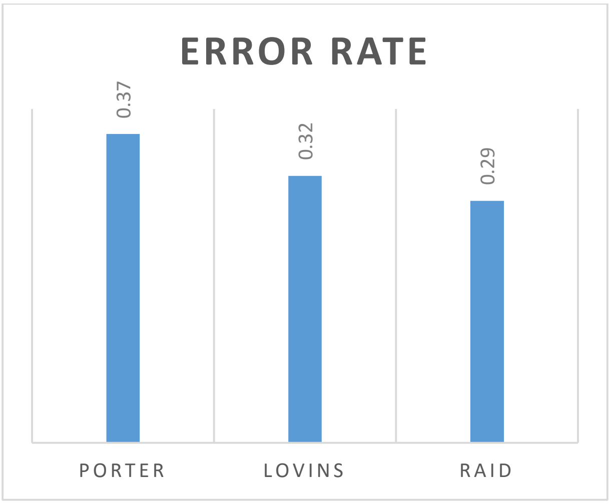


Figure 4. ERROR RATE of PORTER, LOVINS and RAID algorithms

The ERROR RATE is the general measure used by Paice[11] to evaluate the accuracy of a stemmer.
According to this value, the best stemmer would have the lowest ERROR RATE value compared to the rest. So, if we take this measure as a general indicator of performance accuracy, we would have to conclude that RAID is a better stemmer than PORTER and LOVINS. Consequently, these stemmers generate more errors than our approach.

According the results provided in the exprimental phases, we note that an important part of the noise in the results is caused by the absence of some suffixes and certain rules of transformations allowing a good stemming. Our algorithm is able to detect compound words and transform them by minimizing the noise factor.
Comparing our RAID stemmer to the other approaches (LOVINS and PORTER stemmer), we find that the new stemmer not only performs better than PORTER and LOVINS approaches but also it is more accurate than other stemmers. In fact, the differences in error rate values are so important. Regarding the stemmer strength, RAID is lighter than the LOVINS stemmer. This is beneficial for the indexing task since this would improve precision.
In the stemming process the number of retrieved documents increases, because the stem of a term can represents a large concept semantically equivalent than the original term. When the document indexing system uses our new RAID stemmer we perceive an improvement in indexing effectiveness compared to the PORTER and LOVINS stemmers.

This improvement can be explained by three main factors:

- the consideration of compound words.
- the addition of the missing suffixes.
- taking into account the doubling and ignored transformation rules.

VI. Conclusion and future work

The objective of this work is to make contributions to the Stemming problematic for better indexing of unstructured documents. As a solution to this problem, we propose a new algorithm to detect the maximum of relevant words. Indeed, we have developed a first module for processing compound words, and a second one for detecting suffixes that were not taken into consideration by the most algorithms in literature. Our RAID algorithm enabled via the transformation phase,detects and remove suffixes which have not been treated either by the main algorithms such as PORTER and LOVINS. We have experienced our algorithm on a standard basis of terms and on medical corpus. The results were interesting and showed that our algorithm is more efficient than PORTER and LOVINS algorithms.

As perspectives for our approach, we propose further study of irregular verbs, which is not currently taken into consideration by the most of the algorithms in the literature. Also, we intend to improve the basis of the terms of compound words, by adding other suffixes to the English language in order to standardize the algorithm for the treatment of different corpus.

References

[1] M. N. Omri, "Effects of Terms Recognition Mistakes on Requests Processing for Interactive Information Retrieval," International Journal of Information Retrieval Research (IJIRR), vol. 2, no. 3, pp. 19-35, 2012.

[2] A. Kouzana, K. Garrouch, M. N. Omri, "A New Information Retrieval Model Based on Possibilistic Bayesian Networks," International Conference on Computer Related Knowledge : (ICCRK'2012), 2012.

[3] M. Alia, T. Nawal and L. Mourad, "Utilisation d'un module de racinisation pour la recherche d'informations en," INFØDays, pp. p.26-28, 2008.

[4] J. Savoy, "Searching strategies for the Hungarian language," Inf. Process. Manage., p. p 310–324, 2008.

[5] D. Sharma, "Stemming Algorithms: A Comparative Study and their Analysis," International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868, vol. 4, no. 3, pp. 7-12, 2012.

[6] W. Chebil, L. F. Soualmia, M. N. Omri, S. J. Darmoni, "Indexing biomedical documents with a possibilistic network," Journal of the Association for Information Science and Technology, vol. 66, no. 2, 2015.

[7] W. Chebil, L. F. Soualmia, M. N. Omri, S. J. Darmoni, "Extraction possibiliste de concepts MeSH à partir de documents biomédicaux," Revue d'Intelligence Artificielle (RIA), no. 6, pp. 729-752, 2014.

[8] F. Naouar, L. Hlaoua, M. N. Omri, "Possibilistic Model for Relevance Feedback in Collaborative Information Retrieval.," International Journal of Web Applications (IJWA), vol. 4, no. 2, 2012.

[9] P. Majumder, M. Mitra, S. K. Parui, G. Kole, P. Mitra and K. Datta, "YASS: Yet another suffix stripper".," ACM Transactions on Information Systems, 2007.

[10] G. G. David A. Hull, "A detailed analysis of english stemming algorithms," Xerox Research and Technology, 1996.

[11] C. Paice, "An evaluation method for stemming algorithms," In Proceedings of the 7th, pp. p 42-50, 1994.

[12] Paice and D. Chris, "Another stemmer," SIGIR Forum 24, pp. p 56-61, 1990.

[13] D. Nakache, E. Métais and A. Dierstein, "EDA : algorithme de désuffixation du langage médical," Revue des Nouvelles Technologies de l'Information, pp. p 705-706, 2006.

[14] R. Krovetz, "Viewing morphology as an inference process," R. Korfhage et al., Proc. 16th ACM SIGIR Conference, Pittsburgh, pp. p 191-202, 1993.

[15] M. PORTER, " An algorithm for suffix stripping," Program: electronic library and information, pp. p 211-218, 2006.

[16] M. F. PORTER, "An Algorithm for Suffix Stripping," The journal Program, pp. pp.130-137, 1980.

[17] B. A. K. Wahiba, "A NEW STEMMER TO IMPROVE INFORMATION," International Journal of Network Security & Its Applications (IJNSA), pp. p.143-154, 2013.

[18] J. B. LOVINS, "Development of a stemming algorithm," Journal of Mechanical Translation and Computational Linguistics, pp. pp. 22-31, 1968.

[19]M.N. Omri. "System interactif flou d'aide à l'utilisation des dispositifs techniques : Le Système SIFADE ". PhD, Thèse de l'Université Pierre et Marie Curie, 1994.

[20] M.N Omri, I. Urdapilleta, J. Barthelemy, B. Bouchon-Meunier, C.A. Tijus. "Semantic scales and fuzzy processing for sensorial evaluation studies". Information Processing And Management of Uncertainty In Knowledge-Based Systems (IPMU'96). 715-719, 1996.

[21] M.N. Omri & N. Chouigui. "Measure of similarity between fuzzy concepts for identification of fuzzy user requests in fuzzy semantic networks». International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems. 9(6), 743-748,2001.

[22] M.N. Omri & N. Chouigui. Linguistic Variables Definition by Membership Function and Measure of Similarity. Proceedings of the 14th International Conference on Systems Science 2, 264-273, 2001.

[23] M.N. Omri."Possibilistic pertinence feedback and semantic networks for goal extraction", Asian Journal of Information Technology. 3(4), 258-265, 2004.

[24] M.N. Omri. "Relevance feedback for goal's extraction from fuzzy semantic networks", Asian Journal of Information Technology. 3(6), 434-440, 2004.

[25] M.N. Omri, T Chenaina. "Uncertain and approximate knowledge representation to reasoning on classification with a fuzzy networks based system". IEEE International Fuzzy Systems Conference Proceedings. FUZZ-IEEE'99. 3, 1632-1637, 1999.

[26] M.N. Omri. Pertinent Knowledge Extraction from a Semantic Network: Application of Fuzzy Sets Theory. International Journal on Artificial Intelligence Tools (IJAIT). 13(3), 705-719, 2004.

[27] K. BOUKHARI et M. N. OMRI, «SAID : A new Stemmer Algorithm to Indexing Unstructured Document,» The International Conference on Intelligent Systems Design and Applications (ISDA), 2016.

[28] W. chebil, L. F. Soualmia, M. N. Omri et S. J. Darmoni, «BNDI : a Bayesian Network for biomedial Documents Indexing with the MeSH thesaurus,» International Conference on Reasoning and Optimization in Information Systems, 2013.

[29] M. Sendi, M.N Omri, 'Biomedical Concepts Extraction based Information Retrieval Model: application on the MeSH', International Conference on Intelligent Systems Design and Applications (ISDA), pp. 1-6, 2016.

[30] F. Fkih et M. N. Omri, «IRAFCA: An O(n) Information Retrieval Algorithm based on Formal Concept Analysis,» KNOWLEDGE AND INFORMATION SYSTEMS, pp. 1-32, 2015.

[31] F. Naouar, L. Hlaoua et M. N. Omri, «Collaborative Information Retrieval Model Based on Fuzzy Confidence Network,» JOURNAL OF INTELLIGENT AND FUZZY SYSTEM, pp. 1-11, 2015.

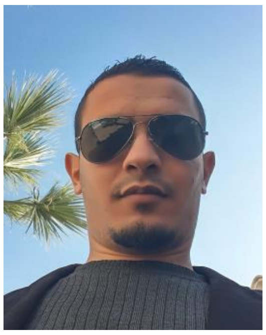
[32] A. Elbahi, M. N. Omri et M. A. Mahjoub, «Possibilistic Reasoning Effects on Hidden Markov Models Effectiveness,» The 2015 IEEE International Conference on Fuzzy Systems , pp. 1-9, 2015.

[33] F. Fkih et M. N. Omri, «Complex Terminology Extraction Model from Unstructured Web Text Based Linguistic and Statistical Knowledge» IJIRR: International Journal of Information RetrievalResearch, pp. 1-18, 2013.

[34] Dawson J. «Suffix removal for word conflation». In Bulletin of the Association for Literary & Linguistic Computing. vol. 2(3), pp. 33-46, 1974.

[35] F. Naouar, L. Hlaoua et M. N. Omri, «Possibilistic Information Retrieval Model based on Relevant Annotations and Expanded Classification,» 22nd International Conference on Neural Information Processing, pp. 1-10, 2015.

Author Biographies



Kabil BOUKHARI He received his Bachelor's degree in computer science, and Master of Automatic reasoning system degrees from Faculty of Sciences of Monastir, TUNISIA, in 2013. He is PhD student and a member of MARS (Modeling of Automated Reasoning Systems) Research Unit. His research interests include web information retrieval and indexing systems.

Mohamed Nazih OMRI received his Ph.D. in Computer Science from Jussieu University, in 1994. He is a Professor in computer science at Monastir University. From January 2011, he served as the Director of MARS (Modeling of Automated Reasoning Systems) Research Unit.
His group conducts research on Approximate reasoning, Fuzzy logic, Modeling of complex systems, web information retrieval, Bayesian and Semantic Networks.