A NEW STEMMER TO IMPROVE INFORMATION RETRIEVAL

Wahiba Ben Abdessalem Karaa

University of Tunis. Higher Institute of Management, Tunisia
RIADI-GDL laboratory, ENSI, National School of Computer Sciences. Tunisia
wahiba.abdessalem@isg.rnu.tn

ABSTRACT

A stemming is a technique used to reduce words to their root form, by removing derivational and inflectional affixes. The stemming is widely used in information retrieval tasks. Many researchers demonstrate that stemming improves the performance of information retrieval systems. Porter stemmer is the most common algorithm for English stemming. However, this stemming algorithm has several drawbacks, since its simple rules cannot fully describe English morphology. Errors made by this stemmer may affect the information retrieval performance.

The present paper proposes an improved version of the original Porter stemming algorithm for the English language. The proposed stemmer is evaluated using the error counting method. With this method, the performance of a stemmer is computed by calculating the number of understemming and overstemming errors. The obtained results show an improvement in stemming accuracy, compared with the original stemmer, but also compared to other stemmers such as Paice and Lovins stemmers. We prove, in addition, that the new version of porter stemmer affects the information retrieval performance.

Keywords

Stemming, porter stemmer, information retrieval

1. INTRODUCTION

Stemming is a technique to detect different inflections and derivations of morphological variants of words in order to reduce them to one particular root called stem. A word's stem is its most elementary form which may or may not have a semantic interpretation. In documents written in natural language, it is hard to retrieve relevant information. Since the Languages are characterized by various morphological variants of words, this leads to mismatch vocabulary. In applications using stemming, documents are represented by stems rather than by the original words. Thus, the index of a document containing the words "computing", "compute" and "computer" will map all these words to one common root which is "compute". This means that stemming algorithms can considerably reduce the document index size, especially for highly inflected languages, which leads to important efficiency in time processing and memory requirements.

First researches about stemming have been done in English. This language has a relatively simple morphology. A diversity of stemming algorithms have been proposed for the English language such as Lovins stemmer [1], Paice/Husk stemmer [2],

2. PORTER STEMMER

Porter stemmer was developed by Martin Porter in 1980 at the University of Cambridge [4]. Porter's algorithm is applied in many fields as a pre-processing step for the indexing task; its main use is as part of a term normalization process that is usually done when setting up an Information retrieval system. The Porter stemmer is actually the most commonly used of all the stemmers. It showed improvements in retrieval performance and in other fields such as classification, clustering, spam filtering…It becomes the most popular and the standard approach of stemming.

The stemmer is based on the idea that the suffixes in the English language are mostly built of a combination of smaller and simpler suffixes; for instance, the suffix "fullness" is composed of two suffixes "full" and "ness". Thus, Porter stemmer is a linear step stemmer; it applies morphological rules sequentially allowing removing affixes in stages.

Specifically, the algorithm has five steps. Each step defines a set of rules. To stem a word, the rules are tested sequentially; if one of these rules matched the current word, then the conditions attached to that rule are tested. Once a rule is accepted; the suffix is removed and control moves to the next step. If the rule is not accepted then the next rule in the same step is tested, until either a rule from that step is accepted or there are no more rules in that step and hence the control passes to the next step. This process continues for all the five steps; in the last step the resultant stem is returned by the stemmer. The whole algorithm can be resumed by the following activity diagram (Figure 1):

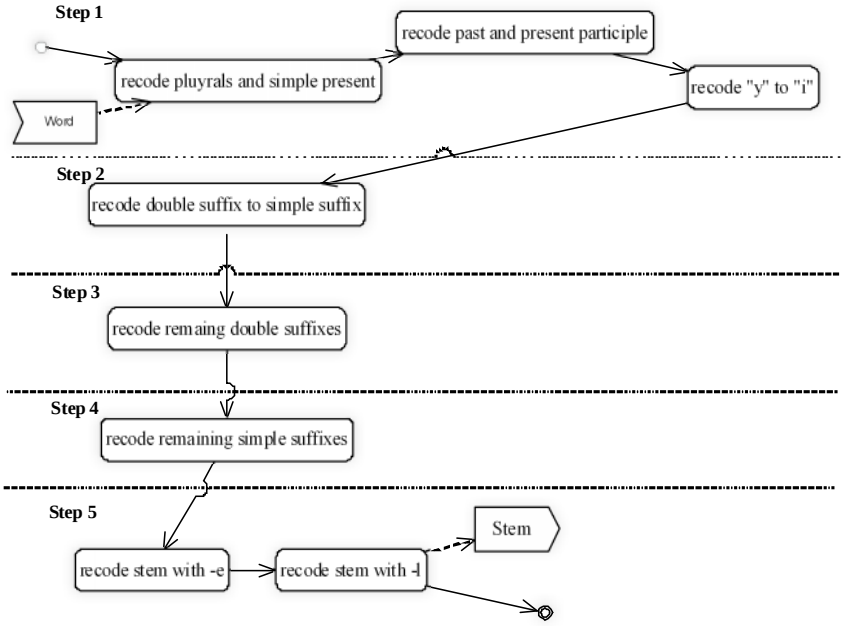

Figure 1. The steps of Porter Stemming Algorithm

Words' suffixes are removed step by step. The first step of the algorithm handles plurals, past participles, present participles, and transforms a terminal "y" to an "I". For example: *"generalizations"* is converted to *"generalization"*, *"agreed"* to *"agree"*, and *"happy"* to *"happi"*. The second step deals with double suffixes to single ones. For example:

"generalization" is converted to *"generalize"*, *"oscillator"* to *"oscillate"*. The third step removes other double suffixes not handled in the previous step such as: *"generalize"* is changed into *"general"*.

The fourth step removes remaining suffixes such as: *"general"* is transformed into *"gener"*, *"oscillate"* to *"oscill"*.

The fifth step treaties stems ending with –e, and treats words ending in double consonant. For example: *"attribute"* is recoded *"attribut"*, *"oscill"* is converted to *"oscil"*.

3. PORTER STEMMER ERRORS

Although Porter stemmer is known to be powerful, it still faces many problems. The major ones are Overstemming and Understemming errors. The first concept denotes the case where a word is cut too much which may lead to the point where completely different words are conjoined under the same stem. For instance, the words *"general"* and *"generous"* are stemmed under the same stem *"gener"*. The latter describes the opposite case, where a word is not cut enough. This can lead to a situation where 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.

For example, Porter stemmer does not treat irregular verbs: *"bought"* remains *"bought"*, and *"buy"* is stemmed *"bii"*, whereas normally the two words have the same stem *"buy"*. Irregular plural nouns are not handled by the stemmer: words ending with –men are the plural of words ending with –man. Porter stemmer makes other errors concerning the terminal –e. For instance, *"do"* is stemmed *"do"* and *"does"* is stemmed *"doe"*. Many exceptions are not controlled: verb conjugation, possessive nouns, irregular comparative and superlative forms (e.g. good, better, best), etc. Moreover, more than 5000 suffixes are not handled by Porter such as –ativist, -ativistic, -ativism, atavistically, –ship, –ist, –atory, –ingly, got, gotton, ound, ank, unk, ook, ept, ew, own, etc. This would decrease the stemming quality, since related words are stemmed to different forms. In an information retrieval context, such cases reduce the performance since some useful documents will not be retrieved: a search for *"ability"*, for example, will not return documents containing the word *"able"*. This would decrease the efficiency of diverse systems applying Porter stemmer.

4. CONTRIBUTION: THE NEW PORTER STEMMER

In order to improve Porter stemmer, we studied the English morphology, and used its characteristics for building the enhanced stemmer. The resultant stemmer to which we will refer as "New Porter" includes five steps (Figure 2):

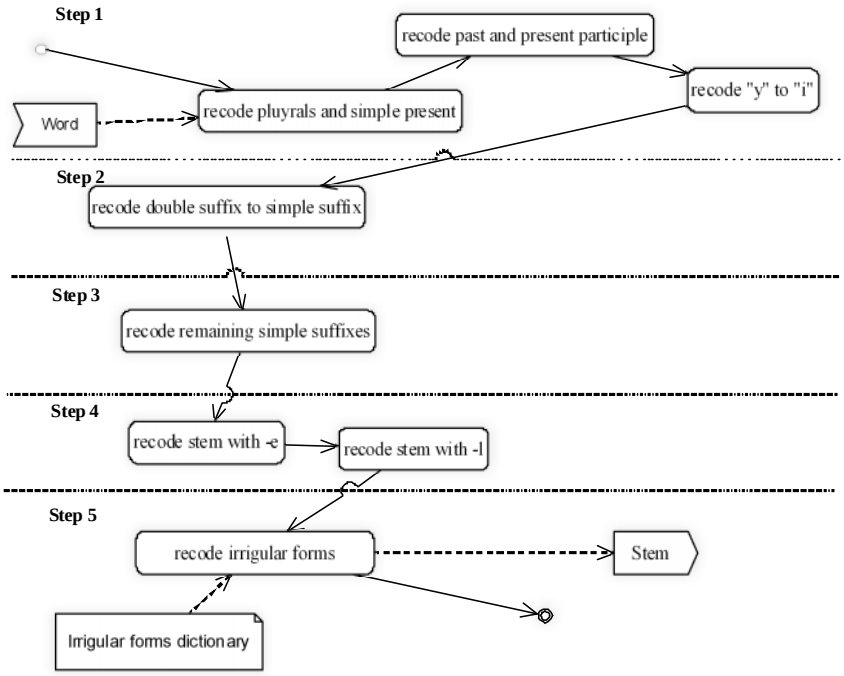

Figure 2. The steps of the new Porter Stemming Algorithm

The first one as in Porter stemmer handles inflectional morphology (plural, verb conjugation, etc.). The second step treats derivational morphology, it maps complex suffixes (suffixes compound of more than one suffix) to a single suffix from which they were derived (e.g. transform the suffix –istic to –ist). The third step deletes simple suffixes (uncompounded suffixes). The fourth step defines a set of recoding rules to normalize stems. The last step treats irregular forms that do not follow any pattern.

In the following, we group in different classes all exceptional cases that are not handled by the original stemmer. For each case, we suggest rules and how many words are concerned by these rules.

4.1 Class 1

In this class we gather all exceptions related to verb conjugation and pluralisation.
Porter stemmer ignores irregular forms. These forms can be categorized into two types: forms that do not follow any pattern; for instance "bought" the past participle of "buy". To handle these cases, we inserted a dictionary in Step 5 containing a list of common irregular forms. The second category concerns forms that follow a general pattern. For instance, words ending in –feet are plural form of words ending in –foot. We propose rules to handle this category. Table. 1 presents the proposed rules and results when applying the two approaches. We give also the number of unhandled words for each category:

Table 1. Handling words belonging to class 1.

Category	Original words	Results using Porter	Rules	Results using New Porter	Number of words
Words ending in -feet are the plural form of words ending in -foot	clubfoot /clubfeet	clubfoot/clubfeet	feet➔-foot	clubfoot/clubfoot	9 words
Words ending in -men are the plural form of words ending in -man	drayman/ draymen	drayman/ draymen	men➔-man	drayman/ drayman	427 words
Words ending in -ci are the plural form of words ending in -cus	abacus/ abaci	abacus/ abaci	ci➔-cus	abacu/ abacu	35 words
Words ending in -eaux are the plural form of words ending in -eau	plateau/ plateaux	plateau/ plateaux	eaux➔-eau	plateau/ plateau	29 words
Words ending in -children are the plural form of words ending in -child	child/children	child/children	children➔-child	child/child	6 words
Words ending in -wives are the plural form of words ending in -wife	farmwife/ farmwives	farmwif/farmwiv	-wives➔-wife	farmwif/farmwif	12 words
Words ending in -knives are the plural form of words ending in -knife	knife/ knives	knif/ kniv	-knives➔-knife	knif/ knif	5 words
Words ending in –staves are the plural form of words ending in -staff	flagstaff/ flagstaves	flagstaff/ flagsttav	-staves➔-staff	flagstaff/ flagstaff	7 words
Words ending in -wolves are the plural form of words ending in -wolf	werewolf/ werewolves	werewolf/ werewolv	-wolves➔-wolf	werewolf/ werewolf	4 words
Words ending in -trices are the plural form of words ending in -trix	aviatrix/ aviatrices	aviatrix/ aviatric	-trices➔-trix	aviatrix/ aviatrix	18 words
Words ending in -mata are the plural form of words ending in -ma	chiasma/ chiasmata	chiasma/ chiasmata	-mata➔-ma	chiasma/ chiasma	108 words

Table 1. (Continued)

Category	Original words	Results using Porter	Rules	Results using New Porter	Number of words
Words ending in -ei are the plural form of words ending in -eus	clypeus/ clypei	clypeu/ clypei	-ei➔-eus	clypeu/clypeu	26 words
Words ending in -pi are the plural form of words ending in -pus	carpus /carpi	carpu /carpi	-pi➔-pus	carpu /carpu	17 words
Words ending in -ses are the plural form of words ending in -sis	analysis/analyses	analysi/analys	-sis➔-s	analys/analys	492 words
Words ending in -xes are the plural form of words ending in -xis	praxis/praxes	praxi/prax	-xis➔-x	prax/prax	32 words

Table 1 shows many cases that are not taken into account by Porter. The proposed rules handle about 1200 exceptional cases that were ignored by the original algorithm.

4.2 Class 2

Porter stemmer does not conflate verbs ending in –s (not –ss) with their participle forms. The stem of the infinitive form is obtained by removing the terminal –s. For the past or present participle, Porter stemmer removes respectively the suffixes –ed and –ing and keeps the terminal –s. Table 2 presents the proposed rules to handle this category and results when applying the two approaches.

Table 2. Handling words belonging to class 2.

Original words	Results using Porter	Rules	Results using New Porter	Number of words
focus/focuses/focused/ focusing	focu/focus/focus/focus	-sed and !(-ssed)➔s -sing and !(-ssing)➔s	focu/focu/focu/ focu	28 verbs
chorus/choruses/ chorused /chorusing	choru/chorus/chorus /chorus		choru/choru/chor/ choru	

4.3 Class 3

Porter stemmer does not conflate words ending in –y and that do not contain a vowel with their derived form. In fact, Porter defines a recoding rule that transforms an –y terminal to –i only if the word contains a vowel. Another rule is defined to handle words ending in –ies which is *ies-> i.* This will conflate carry-carries, marry-marries, etc. However, words such as try-tries-tried are not

conflated since the stem does not contain a vowel. Similarly, verbs ending in –ye are not handled by Porter stemmer. In fact, when a verb ends in –ye, the terminal –e is removed in the last step and hence, the –y is not replaced by –i. To stem the past or present participle of this verb, the suffix –ed or –ing will be removed in the first step, leaving the stem with an –y terminal which will be replaced by –i in the next step and hence, the infinitive form of the verb, its past and present participle are not conflated. To handle these cases, we propose to eliminate the rule defined in the first step that transforms a terminal –y to –i if it contains a vowel. Table 3 presents results when applying the two approaches on a set of words belonging to this category.

Table 3. Handling words belonging to class 3.

Original words	Results using Porter	Results using New	Number of words
cry/cries/cried/crying	cry/cri/cri/cry	cry/cry/cry/cry	About 20 verbs
dye/dyes/dyed/dying	dye/dyes/dyed/dying	dy/dy/dy/dy	

4.4 Class 4

Porter stemmer makes errors concerning verbs ending in a double consonant and their derivations. Thus, if the stem of the present or past participle form ends in a double consonant and that the consonant is other than ‘l’, ‘s’ or ‘z’, then the stemmer removes a letter and keeps the stem with a single consonant. In the last step, the stemmer removes a consonant only for words ending in –ll and for which the m value is greater than 1. This will cause some problems. For instance, "ebbed" is stemmed to "eb". However, "ebb" is stemmed to "ebb". Hence, the two words are not conflated. Exceptional cases are all verbs ending in double consonant other than –l, –s and –z. Verbs ending in –z that double the –z to form the present or past participle are also not treated by Porter stemmer. Thus, these verbs get their past or present participle by doubling the terminal –z. Consequently, these verbs will not be conflated with their infinitive form. For instance, "whizzed" the past participle of "whiz" is stemmed to "whizz"; "whiz" is kept unchanged and hence "whiz" and "whizzed" are not conflated. To resolve these problems, we propose to redefine the recoding rule of the last step. Initially, the rule deletes a double consonant if the consonant in question is –l and that the m value of this stem is greater than 1. The rule will be modified in the way that it deletes the consonant of all the stems ending in a double consonant. For words ending in –ll, It removes a consonant if the stem has an m value greater than 1. Table.4 presents results when applying the two approaches on a set of words belonging to this category.

Table 4. Handling words belonging to class 4

Original words	Results using Porter	Results using New Porter	Number of words
ebb/ebbed/ebbing	ebb/eb/eb	eb/eb/eb	160 verbs
add/added/adding	add/ad/ad	ad/ad/ad	
staff/staffed/staffing	staff/staf/staf	staf/staf/staf	
spaz/spazzes/spazzed	spaz/spazz/spazz	spaz/spaz/spaz	
whiz/whizzes/whizzed	whiz/whizz/whizz	whiz/whiz/whiz	

4.5 Class 5

Porter stemmer does not treat present or past participle derivations. For instance, 'studiedly' is stemmed to 'studiedli' however 'study' is stemmed to 'studi'. Hence, the two forms are not conflated. Table.5 presents the proposed rules to handle this category and results when applying the two approaches on a set of words belonging to this category:

Table 5. Handling words belonging to class 5

category	Original words	Results using Porter	Rules	Results using New Porter	Number of words
Words ending in –iedly or –iedness are related to word ending in -ied	study/studied/ studiedness/ studiedly	studi/studi/studied/studiedli	-ly➔-ied -ss➔-ied	study/study/study/study	13 words
Words ending in –edly or –edness are related to word ending in –ed	amaze/amazed /amazedly/ amazedness	amaz/amaz/ amazedli/ amazed	-ly➔-ed -ss➔-ed	amaz/amaz/amaz/ amaz	439 words
Words ending in –ingly or –ingness are related to word ending in -ing	amaze/amazing /amazingly/ amazingness	amaz/amaz/ amazingli/ amazing	-ly➔-ing -ss➔-ing	amaz/amaz/amaz/ amaz	543 words

4.6 Class 6

Porter stemmer ignores many suffixes such as –est, –ist, –tary, –tor, –sor, –sory, –nor, –ship, –acy, –ee, etc. We propose new rules to handle these suffixes. Many other compound suffixes are also ignored by Porter stemmer. To deal with this problem, we propose to generate all possible compound suffixes derived from each suffix. For instance, suffixes derived from –ate, –ative, –ativist, –ativistic, –ativism, etc. These suffixes will be then mapped to a common suffix (the suffix from which all suffixes were derived). The proposed rules handle an important number of exceptions (more than 5000 exception). In the following we present an extract of the proposed rules to handle these suffixes:

Table 6. Handling words belonging to class 6 (an extract)

Suffix	Proposed rule	Number of words ending in the suffix
- atization	- atization ->-ate	1
- atist	- atist ->-ate	20
- atism	- atism ->-ate	29
- atic	- atic ->-ate	229
- atical	- atical ->-ate	30

The new porter stemmer is implemented. In what follows, we propose to evaluate this stemmer using a standard method inspired by Paice [5].

5. EXPERIMENTAL EVALUATION

5.1 Paice's evaluation method

The development of stemmers aimed to improve information retrieval performance by transforming morphologically related terms to a single stem. This infers that an efficacious stemmer should conflate only pairs of words which are semantically equivalent. The problem is how the program will judge when two words are semantically equivalent. Paice [5] proposed a solution to provide an input to the program in the form of grouped files. These files contain list of words, alphabetically sorted and any terms that are considered by the evaluator to be semantically equivalent are formed into concept groups. An ideal stemmer should stem words belonging to the same group to a common stem. If a stemmed group includes more than one unique stem, then the stemmer has made understemming errors. However, if a stem of a certain group occurs in other stemmed groups, the stemmer has made overstemming errors. This allows the computation of the Overstemming and Understemming Indexes (*UI* and *OI*) and their ratio, the stemming weight (*SW*) for each stemmer.

The Understemming and Overstemming Indexes are metrics of specific errors that occur during the implementation of a stemming algorithm. According to these metrics, a good stemmer should produce as few understemming and overstemming errors as possible. However, they cannot be considered individually during results analysis. To determine the general relative accuracy of the stemmers, Paice defines a measure, called Error Rate Relative to Truncation *(ERRT)*. It is useful for deciding on the best overall stemmer in cases where one stemmer is better in terms of understemming but worse in terms of overstemming. To calculate the *ERRT*, a baseline is used. It is obtained by performing the process of length truncation and reducing every word to a given fixed length. Paice estimates that length truncation is the crudest method of stemming, and he expects any other stemmer to do better. To do so, Paice proposed to determine values of *UI* and *OI* for a series of truncation lengths. This defines a truncation line against which any stemmer can be assessed. Any reasonable stemmer will give an (*OI, UI)* point *P* between the truncation line and the origin. The further away the point is from the truncation line, the better the stemmer is. The *ERRT* is obtained by extending a line from the origin *O* through the (*OI, UI)* point *P* until it intersects the truncation line at *T*, *ERRT* is then defined as: *ERRT= length (OP)/length (OT)*
To apply the Paice evaluation method, lists of grouped word files are required. We used two word lists downloaded from the official website for Paice et Hooper [6]. The first list (Word List A) was initially used by Paice [5] contains about 10000 words. The sample of words was taken from document abstracts from the CISI test collection which is concerned with Library and Information Science. The second list (Word List B) refers to a larger grouped word set (about 20000) compiled from word lists used in Scrabble word checkers.

In this work, we make tests initially with the Porter's original stemmer and the improved version of the stemmer in order to evaluate our approach. We also ran tests with the Paice/husk and Lovins stemmers. The results of these tests are presented in Table 7.

Table 7. Stemming results using the two versions of the stemmers

	Word List A				Word List B			
	UI	OI	SW	ERRT	UI	OI	SW	ERRT
New Porter	0.1882	0.0000518	0.0002753	0.59	0.1274	0.0000441	0.0003464	0.44
Porter	0.3667	0.0000262	0.0000715	0.75	0.3029	0.0000193	0.0000638	0.63
Paice/Husk	0.1285	0.0001159	0.0009020	0.57	0.1407	0.0001385	0.0009838	0.73
Lovins	0.3251	0.0000603	0.0001856	0.91	0.2806	0.0000736	0.0002623	0.86

5.2 Discussion

Comparing Porter stemmer to New Porter, the relative values of indexes are summarized as follows:

UI (Porter)> *UI* (New Porter)
OI (New Porter)> *OI* (Porter)
ERRT (Porter)> *ERRT* (New Porter)

The value of understemming for Porter indicates that it leaves much more words understemmed than New Porter. For instance words such as "ability" and "able" are not conflated when using Porter stemmer which is not true for New Porter. This will reduce Understemming Index and conflate much more related words than Porter especially when the two word lists contain many words ending in suffixes that are not treated by the original stemmer. Hence, the *UI* value is improved by the proposed approach. On the other hand the *OI* value for New Porter is higher than that of the original stemmer. This is the consequence of the New Porter which removes an important number of suffixes affecting a lower number of words. In fact, these rules tend to improve the *UI* value which hurt *OI* and generate more overstemming errors.

For a more detailed analysis, we analyse the structure of word list A and Word List B. We find that an important number of words are related to the derivational morphology. Porter ignores many derivational suffixes such as –est, –ship, –ist, –tor, –ionally, –antly, –atory, etc. All of the words ending with these suffixes are not conflated with other related forms in the original version of the stemmer. This problem was resolved in the stemmer proposed version. The proposed stemmer handles very well derivations and inflections.

The *ERRT* is the general measure used by Paice (1994) to evaluate the accuracy of a stemmer. According to this value, the best stemmer would have the lowest *ERRT* value compared to the rest. So, if we take *ERRT* as a general indicator of performance accuracy, we would have to conclude that New Porter is a better stemmer than Porter. This means that the (*OI, UI)* point of New Porter is farther than the (*OI, UI)* point of Porter from the truncated line. Consequently, Porter stemmer generates more errors than the New Porter stemmer. We state also a large improvement in the *ERRT* values: about 27% for word list A and about 43% for Word List B. This means that the proposed approach performs much better than the original version.

Comparing our stemmer to the other approaches (Lovins and Paice/HUSK stemmer), we find that the new stemmer not only performs better than the original version but also it is more accurate than Paice/HUSK and Lovins stemmers. In fact, the differences in error rate values (*ERRT*) are so important (about 66% with Paice and 95% for Lovins). Regarding the stemmer strength, New Porter is lighter than the Paice/Husk stemmer since it has a lower *SW* value. This is beneficial for the information retrieval task since this would improve precision. Hence, less useless information is retrieved.

5.3 Evaluation in Information Retrieval

The new stemming algorithm described in the previous section is evaluated in textual retrieval field. We implemented a traditional document retrieval system to retrieve a list of documents ranked by their relevance to a query. The system is based on VSM [7], and tf-idf weighting [8], [9].
To evaluate our system we used a corpus of 400 MEDLINE (Medical Literature, Analysis, and Retrieval System Online). MEDLINE is a bibliographic database of the National Library of Medicine, enclosing more than 19 million bibliographic articles, accessible via PubMed [10].
For the comparison, we apply the information retrieval method, first without using a stemmer, second using the original Porter stemmer, and finally using the new Porter Stemmer. We used the two well-known metrics, namely recall and precision, calculated as follows:

$$recall = \frac{correct \bigcap retrieved}{correct}$$

$$precision = \frac{correct \bigcap retrieved}{retrieved}$$

A good information retrieval system should retrieve several relevant documents (have a high recall), and it should retrieve few non-relevant documents (have high precision). As shown in Table 8, the document retrieval system, using the same evaluation documents, outperformed using Porter stemmer, compared to document retrieval system without using a stemmer.

Since the stem of a term represents a larger concept than the original term, the stemming process increases the number of retrieved documents. When the document retrieval system uses the new porter stemmer we perceive an improvement in retrieval effectiveness compared to the original Porter stemmer.

Table. 8. Results in information retrieval

Used stemmer in IR	***Precision***	***Recall***
Without Stemmer	0.661	0.671
With original Porter Stemmer	0.732	0.775
With new Porter Stemmer	0.852	0.884

6. CONCLUSION

This paper suggests an improved version of Porter stemmer for English. The stemmer was evaluated using the Paice evaluation method. In these experiments, we used two grouped word lists of 10000 and 20000 words respectively. Promising results are obtained: thus, the New Porter stemmer performs much better than the original stemmer and other English stemmers.
The new stemmer was also evaluated with an information retrieval system. The obtained results are encouraging: the precision and the recall are improved in an information retrieval system using the new version of porter stemmer compared to an information retrieval system using the original version.

We conclude that the New Porter stemmer is relatively a light stemmer and hence it seems to be appropriate for the information retrieval task. To further confirm the robustness of the stemming algorithm, a perspective to this work is to evaluate the New Porter stemmer in other Natural Language processing areas. The New Porter stemmer will be used in order to improve automatic text summarization, document clustering, information extraction, document indexing, Question-Answering systems, etc. The new algorithm can be useful for words 'normalization as well as reducing the space representation.

Since implementations of this algorithm are available in diverse languages, we plan to produce new versions in further languages to process multilingual corpora.

REFERENCES

[1] Lovins, J.B. (1968). Development of a stemming algorithm. Journal of Mechanical Translation and Computational Linguistics, Vol. 11,No. 1 and 2, pp. 22-31.

[2] Paice, C.D. (1990). Another stemmer. SIGIR Forum. Vol. 24, No.3, pp.56-61.

[3] Porter, M.F. (1980). An Algorithm for Suffix Stripping. The journal Program, Vol. 14, No. 3, pp. 130-137.

[4] Porter, M.F. (2006). An algorithm for suffix stripping. Program: electronic library and information systems, Vol. 40 Iss: 3, pp.211 – 218.

[5] Paice, C.D. (1994). An evaluation method for stemming algorithms. In Proceedings of the 7th Annual Intl, ACM-SIGIR Conference, Dublin, Ireland, pp.42–50.

[6] Paice, C., Hooper, R. (2005). Available at: http://www.comp.lancs.ac.uk/computing/research/stemming/Links/program.htm

[7] Salton, G., Wong, A., & Yang, C.-S. (1975). A vector space model for automatic indexing. Communications of the ACM, Vol. 18. No. 11, pp. 613–620.

[8] Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. Information Processing and Management, Vol. 24, No. 5, pp. 513–523.

[9] Turney, P.D., Pantel, P. (2010). From Frequency to Meaning: Vector Space Models of Semantics Journal of Artificial Intelligence Research No. 37, pp. 141-188.

[10] NLM: National Library of Medicine PubMed Internet (June 2012), http://www.ncbi.nlm.nih.gov/pubmed.

Authors

Wahiba Ben Abdessalem: is a professor in the Department of Computer and Information Science at university of Tunis, at Higher Institute of Management. She received the Master Degree in 1992 from Paris III, New Sorbonne, France, and PhD, from Paris 7 Jussieu France in 1997. Her researches interest Natural language processing, document annotation, information retrieval, text mining. She is a member of program committee of several International Conferences: ICCA'2010, ICCA'2011, ICCA'2013, RFIW 2011, ICCAT'2013, ICCRK'2013, ICMAES'2013, … and a member of the Editorial Board of several journals: the International Journal of Managing Information Technology (IJMIT), the journal International journal of Chaos, Control, Modelling and simulation (IJCCMS)…