informationresearch

VOL. 19 NO. 1, MARCH, 2014

A survey of stemming algorithms in information retrieval

Cristian Moral, Angélica de Antonio, Ricardo Imbert and Jaime Ramírez
Escuela Técnica Superior de Ingenieros Informáticos, Universidad Politécnica de Madrid, Spain

Abstract

Background. During the last fifty years, improved information retrieval techniques have become necessary because of the huge amount of information people have available, which continues to increase rapidly due to the use of new technologies and the Internet. Stemming is one of the processes that can improve information retrieval in terms of accuracy and performance.
Aim. This paper provides a detailed assessment of the current status of the stemming process framed in an information retrieval application field by tracing its historical evolution.
Method. Papers presenting the first approaches for stemming were reviewed to extract their main features, benefits and drawbacks. Additionally, papers dealing with stemmers for non-English languages or with some more recent proposals were also consulted and compiled. Finally, experimental papers defining the most well-known methods and metrics aimed at evaluating and classifying stemmers were also taken into account to expose their contributions and results.
Results. Even if not all researchers agree on the benefits and drawbacks of using stemming in an information retrieval process in general terms, many of them agree on its benefits in specific contexts, such as when the language is highly inflective, when documents are short or when there is limited space for storing data. Some researchers also state that the nature of the documents can influence the performance and the accuracy of the stemmer.
Conclusions. Despite many researchers having investigated this field over many years, there are still some open questions, such as how to evaluate a stemmer independently of the information retrieval process, or how much a stemmer improves an information

retrieval application in terms of speed. As a summary, some guidelines are also provided to help readers to determine which is the best stemmer for their needs and the tasks they have to carry out.

CHANGE FONT

Introduction

During recent decades, there has been a huge growth in the volume of data generated around the world. The trend began around fifty years ago when the research community experienced the first explosion of scientific publications coming from many different domains (Luhn, 1957), later boosted by the arrival and the subsequent socialization of the Internet. The need to allow researchers to find information among these big collections of data promoted the implementation of some mechanisms to support the task (Bell and Jones, 1979). Today, these mechanisms are part of the information retrieval process (Baeza-Yates and Ribeiro-Neto, 1999; Manning, Raghavan, and Schütze, 2008), which is a full, extensive, and specialized field of research in information technology.

The main objective of information retrieval is to automatically analyse and treat documents to extract some measures and relevant data allowing users to easily meet, as well as possible, their *information need*. By information need we mean a high level concept that encompasses not only direct questions (e.g., What is stemming?), but also searches for documents referring to a term or set of terms that can be very specific (e.g., stemming) or abstract search expressions reflecting that the user is not sure about what s/he is looking for (e.g., algorithm to find words' roots). These direct questions, terms and expressions are called *queries* and they represent the user's input to the system. The output answer is generally a document title or a set of document titles, which can be ranked according to a matching rate between the documents and the query calculated during the process. This process is a pipeline of independent but complementary tasks that uses as input a *corpus of documents* and a user's query, and returns as output a set of measures showing how well each and every document answers the information need posed by the user.

One of the first steps in the information retrieval pipeline is stemming (Salton, 1971). A stemming algorithm, or *stemmer*, aims at obtaining the stem of a word, that is, its morphological root, by clearing the affixes that carry grammatical or lexical information about the word. In both cases, these affixes do not modify the concept the word is related to as the semantic informality has been proven in the literature, especially in languages that are highly inflective (Popovic and Willett, 1992) and in short documents (Krovetz, 1993), in terms of *recall* and *precision*.

In this paper we provide a review of the evolution and state of the art of stemming algorithms in the last forty-five years, covering not only the functioning of the most well-known and used algorithms for this task (Smirnov, 2008), but also analysing evaluations conducted and results obtained by many researchers through experimentation. We describe which evaluation metrics have been proposed and which of them have allowed the experts to classify and compare different approaches. We complete the survey including some more recent proposals to improve the process. We finally dissect some algorithms that stem in languages different to English, relying on the same principles but adapting the process to the singularities of each of these languages.

Stemming algorithms: purpose

A stemming algorithm, or stemmer, has three main purposes. The first one consists of clustering words according to their topic. Many words are derivations from the same stem and we can consider that they belong to the same concept (e.g., *drive*, *driven*, *driver*). These derivations are generated through appended affixes (prefixes, infixes, and/or suffixes) but, in general, and more specifically in English, only suffixes are considered, as generally prefixes and infixes modify the meaning of the word, and stripping them would lead to errors of bad topic determination (Hull, 1996). Some exceptions to this occur in very *inflective languages* like German and Dutch (Kraaij and Pohlmann, 1994), or in documents belonging to some specific topics, like medicine or chemistry, where prefixes and suffixes maintain the concept of the word. Among these suffixes two types of derivations can be considered (Krovetz, 1993). In the first case, *inflectional* derivations reflect grammatical information, related to the gender, number, case, mood or tense. These derivations do not provoke a change in the *part-of-speech* of the original word (that is the linguistic category of the word, e.g., noun, verb or adjective) nor in its meaning. On the contrary, *derivational* suffixes deal with the creation of new words based on an existing word, with which it can share the meaning or not (e.g., words terminating in *-IZE*, *-ATION*, *-SHIP*). Stripping these suffixes from a derived word allows its stem to be obtained, which is nearly its morphological root, and then thematically related words can be identified by matching their stems.

The second purpose of a stemmer is directly related to the information retrieval process, as having the stems of the words allows some phases of the information retrieval process to be improved, among which we can highlight the ability to index the documents according to their topics, as their terms are grouped by stems (that are similar to concepts) or the expansion of a query to obtain more and more precise results. The expansion of the query allows it to be refined by replacing the terms it contains with their related topics that are also present in the collection, or by adding these topics to the original query. This adaption can be done automatically and transparently to users or the system can propose one or more improved formulations of the query to users letting them decide if any of them is more specific and defines better their information needs. Even if interactive query expansion is better in principle because the user has more feedback on what is happening, typically it cannot be done directly with the result of stemming, as stems usually are not understandable by humans (Voorhees, 1994).

Finally, the *conflation* of the words sharing the same stem leads to a reduction of the dictionary to be taken into account in the process, as the whole vocabulary contained in the input unprocessed collection of documents can be reduced to a set of topics or stems. This leads to a reduction of the space needed to store the structures used by an information retrieval system (like the index of terms-documents) and then also lightens the computational load of the system.

Classical approaches

The process of conflating words has been widely investigated during recent decades and, in general terms, two main approaches have been adopted. The first one is an algorithmic-based approach where no linguistic considerations, such as gender, number or verbal tense, are considered. These stemmers work on the base of a set of rules defining if a suffix has to be removed or not, depending on some conditions. As no linguistic consideration is taken into account, in many cases the resulting stem is not a well-formed word (sometimes it is not even a recognizable word) and then it is empty of meaning. Depending on the task, this can be invisible for users, as the resulting stems are not presented to them (e.g., when indexing documents by concepts) or it can be a drawback, as the user is not able to understand the extra information given by the system (e.g., query expansion). Then, when the undertaken task requires meaningful output stems, a different approach has to be chosen. A linguistic-based approach, where the inner semantic information of the word is exploited to group related words into the same concept independently from their morphology, becomes then the better option. This semantic information can be obtained through dictionaries or thesauri (Miller, 1995), which contain hierarchical, semantic and lexical information about words. This second approach has rapidly diverged into a different field of research called *lemmatization*, which will not be addressed in this paper.

The first stemmer mentioned in the literature is the Lovins Stemmer (Lovins, 1968). The algorithm consists of two steps: elimination of the suffixes and treatment of the remaining stem. The suffix stripping is done by matching the termination of the word with the longest suffix from an ordered list of 294 suffixes and applying one of the twenty-nine associated application rules. After this the remaining stem is treated to solve some linguistic exceptions like *double d* or *double t* endings. This step, called the *recoding phase*, is also based on a set of thirty-five rules that determine if the termination of the remaining stem has to be partially deleted or modified (e.g., *inputting* should be stemmed to *input*, not to *inputt*). Finally, conflation is done with a partial-matching algorithm that groups words whose stems are very close, but not necessarily the same. This permits an increase in the *conflation rate* and it is based on the assumption that two stems that are mostly equal but have little differences because of the suffix stripping process (e.g., *explain* and *explanation* are differently stemmed to *explain* and *explan*) belong to the same concept and should be grouped together. Nevertheless, this, obviously, can increase the errors by conflating words that are not related but mostly share the same stem (e.g., *probe* and *probate* are respectively stemmed to *probe* and *prob*, and then are likely to conflate by partial-matching). To avoid these errors without losing precision or recall, Dawson proposes an adaptation of the Lovins' stemmer (Dawson, 1974) based on two modifications. Firstly, the recoding phase is omitted and only the partial-matching routine is used to conflate words. Secondly, to fill the gap left by the recoding phase, he greatly increases the list of common suffixes up to 1200 to handle a lot more situations that were absent in the Lovins stemmer and were many times solved in the recoding phase.

Even if Lovins stemmer is one of the most widely known stemmers, Porter's stemmer (Porter, 1980) is the most used in information retrieval, probably because of its balance between simplicity and accuracy. Porter defines a five step algorithm applied to every word in the vocabulary. A word is defined as a succession of vowel-consonant pairs $[C](VC)^m[V]$, where C and V are lists of one or more consonants and vowels respectively and m is the measure of the word. The algorithm counts on around sixty rules that are divided into five steps that define the conditions under which they get applied (the term suffix has to match the suffix condition appearing in the rule and the remainder of the term, that is, the initial term without the suffix has to satisfy the condition also indicated in the rule) and how the term is modified to obtain the stem (which suffix replaces the former suffix). As an example, the rule $(m > 1)\ EMENT \rightarrow \varnothing$ indicates that if a term has the suffix *-EMENT* and the measure *m* of the remainder of the term is greater than one (in this case *m*=2), then the suffix *-EMENT* is deleted. Then, according to this rule, the stem of the term *replacement* is *replac*. The algorithm is considered concise enough to avoid using a recoding phase or a partial-matching approach, as the set of rules are executed sequentially to remove complex suffixes by treating them as a set of more simple suffixes that are cleared in every step of the algorithm (e.g., *generalizations* is incrementally reduced to *generalization*, *generalize*, *general* and finally *gener* through four different steps of the algorithm).

Another well-known proposal is the Paice/Husk stemmer (Paice, 1990), which is an iterative algorithm using the same rules and suffixes in every loop. Each rule is divided into five parts, with two of them being optional:

- the suffix, written in inverse order to ease matching with the words' terminations,
- the symbol '*' indicating that the term can be stemmed only one time (optional),
- the number of letters that must be cleared from the termination of the term,
- the string that must be appended to the cleared form of the term (optional),
- the symbol '>' indicating that the term can be treated in the next iteration, or the symbol '.' indicating that the term's final stem has been obtained.

An example of this rule is *sei3y>*, where terms with the termination *-IES* get their three last letters cleared and the letter *-Y* is appended. The obtained word is considered again in the next iteration of the stemming process. For example, the term *flies* would be stemmed to *fly*. This ability to delete some letters and to append new ones, which in practice means replacing part of the stem, is equivalent to the recoding phase, which is indirectly incorporated into the rules themselves.

Finally, Krovetz (1993) has proposed a very simple algorithm which is supposed to cover the three most common inflectional derivations that occur: plural forms, past tense and *-ING* verb forms. In addition, some recoding rules are defined to obtain meaningful words after deleting the suffix, using as a support a dictionary against which the stemmed words are compared.

Even though some new approaches have been proposed in the literature to address the problem of stemming, most of them are based on the classical ones, or are developed to stem words from other non-English languages. In fact, nowadays, many researchers still use Porter's stemmer for many different tasks (Patil and Patil, 2013; Chintala and Reddy, 2013) or have tried to improve it (Ben Abdessalem Karaa, 2013).

Evaluation and comparison

Some metrics have been defined in the literature to determine how well a stemmer behaves. These metrics allow not only measuring the performance (that is the precision and recall) of every single stemmer, but also comparing them in different aspects and, therefore, obtaining some kind of classification.

Stemming errors

The first step in calculating the precision of a stemmer consists of identifying what type of errors can be made, under which conditions they occur and how they affect the result. In stemming, two main errors have been identified, both related to the aggressiveness with which the stemmer clears the terminations of the terms. On the one hand, when the stemmer deletes only the terminations that are almost certainly a suffix that must be cleared, there is a risk of keeping some more complex suffixes or ambiguous terminations in the remaining stem that should also be stripped to obtain the morphological root of the word. This error is called *under-stemming*, as the stemmer stripping is under the expected level. On the other hand, a stemmer makes an *over-stemming* error when it strips more terminations than it should, clearing parts of the word that belong to the morphological root. When this error occurs, there is a loss of semantic information as part of the morphological root is deleted. Porter goes beyond these definitions and divides this type of error into two cases (Porter, 2001): over-stemming when the suffix stripped provokes a change of the meaning of the stem (that is, the termination cleared is a suffix, but one belonging to the root) and *mis-stemming* when the cleared termination is not a suffix, but part of the root.

Both errors imply a decrease in the performance of the stemmer, but differently. When a word is under-stemmed it becomes more difficult to identify if two words are related based on their morphology, as their obtained stems are mostly equal, but not totally because of a suffix that has not been deleted. The opposite case takes place when over-stemming is done. In this case the problem is that it becomes more possible that two words that are not related but share part of their morphological root are wrongly detected as related because their stems are equal.

Some solutions have been proposed to reduce these errors as much as possible. For over-stemming most of the proposed stemmers set a minimum size constraint on the resulting stem. This means that a term is stemmed only if its stem has a minimum length that typically is two or three letters. By doing this, the algorithm avoids deleting or modifying the termination of a term that matches one of the suffixes of the list but is not actually a suffix. As an example, a rule where the suffix *-IES* is replaced by the suffix *-Y* should impose on the remaining term a size greater than one to avoid stemming the conjugated verb *DIES* to *DY*. Therefore it is essential to define the stemming rules as carefully as possible to consider most cases and then prevent uncontrolled situations (Dawson, 1974; Porter, 1980; Paice, 1990). In the other case, the under-conflation provoked by under-stemming can be palliated by applying partial-match algorithms (Dawson, 1974; Lovins, 1968) that determine that two stems are related if the similarity between their morphology is over a defined threshold. Even if these solutions are useful only in some cases and, sometimes, can introduce new errors, globally they yield good results. Their adoption depends on the required needs for the task: if the conflation is an important aspect, even at expense of the thematic precision of the resulting groups, then solutions against under-stemming will be adopted; conversely, if it is preferable to have perfectly classified groups according to their topic, even if they are little and lack some related documents, then over-stemming errors should especially be addressed. Typically, a trade-off between the two approaches is desired, even if it increases the difficulty of the process.

Classification and evaluation

Based on the errors defined above, a classification has been proposed to label a stemmer depending on which type of errors it most commonly does. The *strength* of a stemmer is defined as the aggressiveness with which the stemmer clears the terminations of the terms and it depends on the rate of over-stemming and under-stemming made by a stemmer. A *strong* (or *heavy*) *stemmer* has a high rate of over-stemming errors, while a *weak* (or *light*) *stemmer* is characterized by having a high under-stemming rate.

Paice (1994) proposes some metrics to evaluate a stemmer regardless of the task carried out: the *under-stemming index* (*UI*), the *over-stemming index* (*OI*), the *stemming weight* (*SW*) and an *error rate relative to truncation* (*ERRT*). An experiment with the Lovins, Porter and Paice/Husk stemmers showed that Paice/Husk has the highest rate of over-stemming and Porter the lowest and, on the contrary, Porter makes more under-stemming errors than the others, with Paice/Husk being the one generating least errors of this type. As Paice considers that the strength of a stemmer can be directly defined by its over-stemming and under-stemming indexes (*SW=OI/UI*), he concludes that Paice/Husk is the strongest stemmer, followed by Lovins which is still considered a strong stemmer, and finally Porter, which is the weakest among the three. This result is supported by Frakes and Fox (2003) who, using an inverse modified Hamming distance measure, also affirm that Paice/Husk is stronger than Lovins, which in turn is greatly stronger than Porter. They also compute the strength of the "S" stemmer, which, as expected, is much weaker than Porter. The "S" stemmer deals only with plural forms and its treatment has been proposed by Harman (1991) as a baseline for evaluation and comparison of stemmers. The simple truncation of a fixed number of letters has also been used in many cases as a baseline algorithm for comparisons (Braschler and Ripplinger, 2004; Paice, 1994). Table 1 summarizes the main features of the presented stemmers and their strength.

Table 1: Summary of classical stemmers features.

Stemmer	**Number of rules**	**Number of suffixes**	**Use of recoding phase**	**Use of partial-matching**	**Strength**	**Use of constraint rules**
Lovins	29	294	35 rules	yes	strong	yes
Dawson	unknown	1200	no	yes	strong	yes
Porter	62	51	no	no	weak	yes
Paice/Husk	115	unknown	no	no	very strong	yes
Krovetz	unknown	5 (-S, -ES, -IES,-ED, -ING)	yes (use of dictionary)	no	very weak	no

However, these metrics do not allow the evaluation of the accuracy of a stemmer, but only the classification of it according to its typical errors. In order to evaluate the accuracy of the stemming process two metrics, introduced by Kent, *et al.* (1955), are used. These metrics have been widely adopted in information retrieval as the standard metrics to evaluate the accuracy of the process. The first metric, recall, reflects the rate of relevant documents that are obtained as an answer to a query, while precision represents the rate of the retrieved documents that are relevant to the query. In other words, when the stemmer tends to group as many related documents as possible, even if other non-related documents are also included in the same group, the recall is high, while if the stemmer builds groups with as few non-related documents within a group as possible, even if some related documents are possibly not included in it, the precision is high. These measures have allowed some authors to link performance in terms of recall and precision to the strength of the stemmer, and then to over-stemming and under-stemming (Frakes and Fox, 2003; Paice, 1994; Harman, 1991). Their conclusions are that strong stemmers will, in general, increase the recall of the results, as they make more probable that two words belonging to the same concept get conflated together, but they also decrease the precision as a higher number of non-related words also get conflated together because of over-stemming. Accordingly, they confirm that weak stemmers are better at correctly conflating related words, thus increasing the precision, but are more likely to avoid conflating related words because of under-stemming, therefore decreasing the recall.

Another way to evaluate an algorithmic-based stemmer is through its conflation rate, also called the *Index Compression Factor (ICF)*, which defines how much the stemming process compresses the input vocabulary and then how much it reduces the storage capacity needed and increases the efficiency of the information retrieval system which has to deal with a thinned dictionary. Porter states that his stemmer reduces the initial vocabulary by a third. Many experiments have proven that the vocabulary compression depends on the strength of the stemmer, as can be seen in Table 2. Lennon, *et al.* (1988), Frakes and Fox (2003), Paice (1994) and Harman (1991) have proved with their experiments that strong stemmers yield a better index compression factor than weaker ones, as they conflate more words.

Table 2: Vocabulary compression (ICF).
(Note: ranges are provided for lennon >*et al.* because experiments are carried out over four collections (see Table 3).)

	Lennon *et al.*	**Frakes**	**Paice**	**Harman**
"S" stemmer	-	1%	-	11.48%
Porter stemmer	from 26.2% to 38.8%	17%	38.90%	28.74%
Lovins stemmer	from 30.9% to 45.8%	29%	44.60%	38.23%
Paice/Husk stemmer	-	33%	51.30%	-

To rank the stemmers according to their performance many experiments have been carried out. Harman (1991) compares, in terms of recall and precision, the behaviour of an "S" stemmer, the Porter stemmer and the SMART system (Salton, 1971) which implements an enhanced version of the Lovins stemmer. In the experiment, an information need is posed through a query and the system returns a ranked list of documents related to the query. Stemming is applied both to the documents and the query, and the evaluation is done with respect to a simple classic ranking technique (returns an ordered list of documents whose vocabulary best matches the terms of the query) and a more sophisticated one using term weighting and query expansion. The results show a non-significant increase of performance when using stemming, but an interesting benefit in terms of space, as the vocabulary is reduced by nearly 40%. The cause of the low improvement in terms of precision is that the stemmer fails (because of over-stemming or under-stemming, depending on the selected stemmer) as many times as it succeeds and that is why the author proposes a selective stemming, where the user, depending on the number and quality of the results and on the task s/he has to carry out, decides whether to use a strong or a weak stemmer. Frakes (1984) also carries out two experiments to evaluate the accuracy of a system based on the Lovins stemmer in terms of a metric called *E* (Van Rijsbergen, 1979) which considers both recall and precision, and concludes that the performance is mostly the same either if the conflation is done manually by experts or if it is done automatically using a stemmer. Another experiment is undertaken by Hull (1996) over five stemmers: "S", Porter, Lovins, Xerox inflectional, and derivational analysers (Xerox, 1994). Even if his results do not prove a high improvement in information retrieval (4-6%), he asserts that stemming is clearly beneficial for recall. In fact, Hull bases this conclusion on the results of three measures: the average precision at eleven points of recall (APR11), the average precision at five to ten documents examined (AP[5-15]) and the average recall at fifty to one hundred and fifty documents examined (AR[50-150]). The first measure is calculated by averaging the precision obtained for each of the defined levels of recall (that is when recall is 0.1, precision is measured, then when recall reaches 0.2 precision is calculated again, and so on) using a set of queries. This metric is one of the most well-known in information retrieval and allows analysing the interdependence between recall and precision. The other two metrics calculate respectively the average precision and the average recall applying stemming to increasing numbers of documents. In his experimentation, Hull notes that both APR11 and AP[5-15] measures remain mostly the same with or without stemming, while AR[50-150] increases when stemming is applied. Krovetz (1993) also confirms that the use of stemming brings a big enhancement in terms of performance and especially of recall when the documents are short.

In addition, Frakes and Fox (2003) propose a similarity measure allowing the determination of similarity between two stemmers by comparing the results they return. This measure allows the easy classification of a new stemmer depending on which is the most similar known-stemmer. The similarity is calculated on the basis of the inverse modified Hamming distance. The experiment carried out by the authors over four stemmers ("S" stemmer, Lovins, Porter, Paice/Husk) concludes that Paice and Lovins stemmers are the most similar, while the Paice and "S" stemmers are the most different. Analysing the similarity measures of all the possible combinations of two stemmers the authors conclude that the algorithms similarity depends on their strength similarity.

Figure 1 sums up the features presented above for four of the main classical approaches for stemming. This summary reinforces the idea that all the features depend on the strength of the stemmers and then on the rate of over-stemming and under-stemming errors.

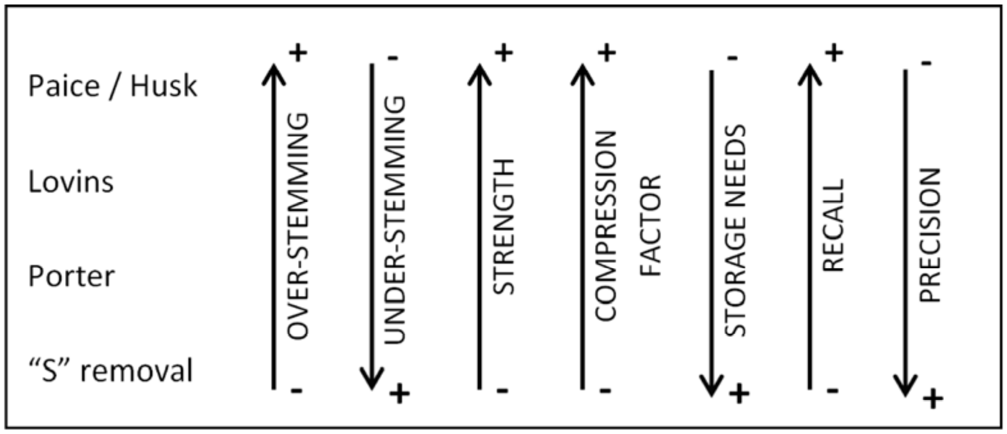

Figure 1: Features summary of classical stemmers.

To evaluate stemmers and obtain these measures a big collection of documents is needed. Over recent years, many collections have been proposed and many of them have become reference sets to test new stemmers and, more generally, to evaluate new approaches to any step of the information retrieval process. Table 3 shows which corpora of documents have been used in the bibliography cited in this paper. In turn, Table 4 lists the most widely known and used collections, and their main features.

Table 3: Corpora used by researchers.

Publication	**Corpora used**
Adam, Asimakis, Bouras and Poulopoulos (2010)	Unknown collection (News articles and e-mail messages)
Braschler and Ripplinger (2004)	CLEF 2001
Chen and Gey (2002)	Agence France Press (383,872 Arabic articles)
Fernández, Díaz, Gutiérrez and Muñoz (2011)	MIKTEX 2.6 Dictionary (Spanish terms list)
Frakes (1984)	Unknown collection(12,000 abstracts about psychology)
Frakes and Fox (2003)	MOBY, Unix spelling dictionary
García Figuerola, Gómez-Díaz, Zazo Rodríguez and Alonso Berrocal (2001)	CLEF
Harman (1991)	CACM, MEDLARS, Cranfield
Hull (1996)	TREC, WSJ
Korenius, Laurikkala, Järvelin and Juhola (2004)	Unknown collection (53,893 articles from Finnish newspapers)
Krovetz (1993)	CACM, Time, NPL, WEST
Kraaij and Pohlmann (1994)	CELEX database

Kraaij and Pohlmann (1996)	Dutch publishers' database (59,608 articles from three regional newspapers)
Larkey, Ballesteros and Connell (2002)	Arabic TREC 2001
Lennon, Pierce, Tarry and Willett (1988)	Cranfield, Brown, INSPEC, NPL
Majumder, Mitra and Pal (2008)	CLEF 2006
Paice (1994)	CISI
Taghva, Elkhoury and Coombs (2005)	Arabic TREC 2001

Table 4: Reference corpora in information retrieval.

Name of the corpus	**Type of data**	**Volume of data**	**Topics of data**	**Source**	**Available at**	**Free**
Brown	Documents	500 documents, 1 million terms	Press, religion, skill and hobbies, popular lore, belles -lettres, government and house organs, learned, fiction and humour	American English texts printed in 1961	Brown Corpus (Archived by WebCite® here)	No
CACM	Abstracts	3204 documents	Engineering	Communications of the ACM journal	CACM Corpus (Archived by WebCite® here)	Yes
CISI	Abstracts	1460 documents	Library Science	Books and scientific journals	CISI Corpus (Archived by WebCite® here)	Yes
CLEF	Articles	2006 edition: 536,678 terms	News	Newspaper in Dutch, English, French, German, Italian and Spanish	CLEF Corpus (Archived by WebCite® here)	No
Cranfield	Abstracts	1400 documents, 139,935 terms (4,632 after stop-words removal)	Aeronautical engineering	Scientific journals	Cranfield Corpus (Archived by WebCite® here)	Yes
INSPEC	Abstracts	10 million documents	Astronomy, electronics, communications, ergonomics, computers and computing, computer science,	Scientific and engineering journals since 1898	Inspec Corpus (Archived by WebCite® here)	No

			control engineering, electrical engineering, information technology and physics			
MEDLARS	Abstracts	1033 documents	Medicine	Scientific journals	Medlars Corpus (Archived by WebCite® here)	Yes
NPL	Abstracts	11,429 documents	Reports of vehicle evaluations	National Physical Laboratory	NPL Corpus (Archived by WebCite® here)	Yes
Reuters-21578	Articles	21,578 documents	News	Reuters	Reuters-21578 Corpus (Archived by WebCite® here)	Yes
Time	Articles	425 documents	News	Time magazine	Time Corpus (Archived by WebCite® here)	Yes
TREC	Articles	≈1.89 million documents	News	Newswire and other sources	Trec Corpus (Archived by WebCite® here)	No
WSJ	Articles	≈30 million of terms	News	Wall Street Journal	WSJ Corpus (Archived by WebCite® here)	No

Some researchers (Krovetz, 1993; Hull, 1996; Harman, 1991) relate the performance of a stemmer to the collection used as input. In fact, they all confirm after their experimentation that stemmers perform better, mainly in terms of precision, when the documents are short (less than 200 words per document). This can be explained because the probability that terms used in the query appear in the same form in the documents decreases as the vocabulary is reduced, and then the conflation obtained by the stemming allows detecting more related terms. This hypothesis also applies for the length of the query.

Stemming in other languages

Although many of the stemmers are aimed at and are evaluated with the English language, some researchers have modified existing approaches or proposed new ones to handle other languages. According to linguistics, languages can be divided into two main categories depending on their morphological structure: *analytic languages* and *synthetic languages*. The second category can be divided, in turn, into three subcategories, namely *agglutinative languages*, *fusional* (or *inflecting*) *languages* and *polysynthetic languages* (Comrie, 1989; Pirkola, 2001; Aikhenvald *et al.*, 2007). However, this classification is an idealization, as all languages belong to more than one category, even if they generally fit better into one of them. In fact, two continuous variables have been proposed to indicate to what extent a language belongs to one type or another, the *index of synthesis* and the *index of fusion* (Whaley, 1997). The index of synthesis describes the extent of morphological synthesis, that is, how much the words of a language are affixed. At one extreme are the most analytic languages (also called *isolating languages*), where all morphemes are free morphemes (language tends to have no morphology at all), while at the other extreme, the most polysynthetic languages tend to have sentences consisting of a single complex word formed by many morphemes. The second variable, the index of fusion, describes how easy it is to split the words into morphemes. On this scale, one extreme would be the agglutinative languages that have words that can be easily separated into clearly identifiable morphemes, while at the other extreme would be the fusional languages, where words are formed by morphemes that are not clearly identifiable. Using these indexes, Table 5 provides the prevailing morphological type of some of the languages for which stemmers have been developed. This diversity of languages and types implies there is also diversity in the problems and challenges that sometimes need to be handled by new non-classical approaches as the classical ones lead to results that are not as accurate and efficient as in English (Kettunen, 2009).

Table 5: Prevailing morphological type of some languages

	Morphological type			
Language	**Analytic/ isolating**	**Synthetic**		
		Agglutinative	**Fusional/inflecting**	**Polysynthetic**
Indonesian (Asian *et al.* 2005)	X			
English (Lovins, 1968; Porter, 1980; Paice, 1990; Krovetz, 1993)		X		
Finnish (Korenius *et al.*, 2004)		X		
Hungarian (Majumder *et al.*, 2008)		X		
Basque (Otxandorena, 2010)		X		
Turkish (Dinçer, 2003; Eryiğit and Adalı, 2004)		X		
Bengali (Majumder *et al.*, 2007)			X	
German (Braschler and Ripplinger, 2004)			X	
Dutch (Kraaij and Pohlmann, 1994)			X	
Portuguese (Orengo and Huyck, 2001)			X	
			X	

French (Moulinier, McCulloh and Lund, 2001)		
Czech (Majumder *et al.*, 2008)	X	
Arabic (de Roeck and Al-Fares, 2000; Larkey *et al.*, 2002; Chen and Gey, 2002; Taghva *et al.*, 2005)	X	
Bulgarian (Majumder *et al.*, 2008)	X	
Slovene (Popovic and Willett, 1992)	X	
Greek (Adam *et al.*, 2010)	X	
Mohawk		X

In fact, even if modern English can be considered an analytic language, it is also a weakly inflective language due to its heritage of Old English, which was a fusional language (Meyer, 2010). This means it is much easier to obtain the stems of the words than in other very isolating languages like Mandarin Chinese or Indonesian. In fact, while stemming in English only cares about suffix stripping, stemming in Indonesian must consider and remove a wider range of affixes, like prefixes, infixes, confixes (combination of prefixes and suffixes), as well as suffixes. Many of the proposed algorithms for stemming Indonesian words use well-known elements used in classical stemmers like dictionaries, lists of rules and recoding (Asian, Williams and Tahaghoghi, 2005).

With respect to agglutinative languages, Finnish (Kettunen, Kunttu and Järvelin, 2005) and Turkish are good examples of highly inflective languages deriving from a complex morphology. In particular, Turkish has approximately 23,000 stems and words are formed depending on their grammatical function, which is indicated by adding suffixes to stems. This results in a theoretically infinite number of words that can be created (Hankamer, 1984). This theoretical infinity of words makes the use of stemming algorithms highly recommended in order to increase the conflation rate and reduce the storage volume (Dinçer, 2003). Due to the morphological complexity of Turkish, typically *morphological analysers* have been proposed as stemmers (Eryiğit and Adalı, 2004). However, morphological analysers are computationally very costly, which is why other approaches have tried to reduce this complexity while trying to obtain an acceptable level of accuracy, for example using n-grams (Ekmekçioglu, Lynch and Willett, 1996) or removing affixes using a set of rules and a list of well-known suffixes, as with Porter's stemmer (Çilden, 2010).

On the other side, fusional languages have a large amount of representatives, as most of the Indo-European languages are of this type. Many of the stemmers for them are based on Porter's approach, as it fits perfectly with their morphological structure. One example is the algorithmic stemmer proposed by Popovic and Willett for the Slovene language (Popovic and Willett, 1992) that uses a list of 5,276 well-known suffixes. This number is much higher than Porter's original one because Slovene is morphologically much richer than English, and then the casuistry, both for inflectional and derivational suffixes, is bigger. The experiments carried out to test this Slovene stemmer show a significant improvement in terms of precision while maintaining recall. In fact, many researchers ensure that the performance of a stemmer increases as the morphological complexity of the language does (Popovic and Willett, 1992; Chen and Gey, 2002; Kraaij and Pohlmann, 1994). To prove it, Popovic and Willett carried out the same experiment using as input the Slovene corpus translated to

English and compared the performance measures obtained. The results of the English experiment showed a non-significant improvement in terms of precision and recall, and then confirmed the relation between performance and morphological complexity of the language. Other more sophisticated approaches have been proposed, always based on the use of suffix removal rules, but improving the result by adapting their application to the peculiarities of the language. This is the case of the stemmer developed by Adam *et al.* (2010) dealing with the Greek language, which is also a morphologically rich language. Moreover, Greek has an added difficulty, as many suffixes depend on the grammatical type of the word within the sentence. That is why the authors propose enriching a typical algorithmic stemmer with an initial *part-of-speech tagging phase* to obtain every word's grammatical category. Then, stripping rules are applied to a subset of the list of possible suffixes, depending on the part-of-speech tag of the term to be stemmed. Due to this, the Greek stemmer proposed yields very good results, as 96.7% of the vocabulary (considering only nouns, adjectives, verbs and pronouns) is stemmed correctly. Analysing the detail of the errors (12.5% of them are because of over-stemming while the rest are caused by under-stemming), the stemmer can be classified as a weak stemmer.

Finally, to the best of our knowledge, no stemming algorithms have been proposed for polysynthetic languages, like Mohawk, Blackfoot or Greenlandic. Probably, in these languages, the fact that words are composed of many morphemes complicates the identification of a single stem per word.

Nevertheless, even if some alternative approaches have been proposed to meet the specificities of certain languages, the Porter algorithm is still one of the most used approaches because of its simplicity, extensibility and adaptability. Moreover, Porter has eased the task by creating *Snowball* (Archived by WebCite® here), a framework to develop new stemmers (Porter, 2001). *Snowball*, which is presented as an easy-to-learn and easy-to-use language, permits an ANSI C or JAVA version of a stemmer to be obtained simply by defining the rules with some lines of code. Currently, stemmers for around twenty different languages have been implemented using *Snowball*, among which are the majority of the agglutinative and fusional languages listed in Table 5.

Recent approaches

To enhance the performance of the stemming and/or to reduce as much as possible the loss of accuracy because of over and under-stemming errors, some researchers have proposed alternative approaches in recent years (Mayfield and McNamee, 2003; Peng, Ahmed, Li and Lu, 2007). Many of them are based on exploiting the context or the domain, either of the term itself or of the document. One example is the work of Xu and Croft (1998), where they propose to consider the co-occurrence of two terms within a collection of documents to determine if they belong to the same concept. If they do, then they must be conflated under the same stem. Their hypothesis is that if two terms are related and belong to the same topic it is highly probable that they will occur together in a given window of words within the documents of the collection. According to this, they propose refining the *equivalence classes of terms* (all the groups of terms that will conflate into the same stem) obtained after applying a heavy stemmer by subdividing or grouping them. This refinement, based on the concept of hierarchical clustering, is done with two algorithms that make use of a metric *em*, based on mutual information theory, that indicates how possible it is to have the same term within a window of words. The first algorithm, called *Connected Component Algorithm (CCA)*, divides an initial equivalence class into subclasses each containing terms with an *em* higher than a defined threshold. This algorithm can affect the effectiveness of the retrieval because of the appearance of *strings* (creation of big classes of equivalence that internally are sparsely connected (Salton, 1989)), which is why the *Optimal Partition Algorithm (OPA)* is used to calculate the best equivalence groups to maximize the performance of the whole system. To achieve this, for every pair of words *(a, b)* the net benefit of conflating them or not is calculated, which is the difference between the measure *em(a, b)*, representing the benefit of conflating, and the constant δ, representing the harm it causes to precision to conflate *a* and *b*. Finally, the best optimal partition is the one that maximizes the sum of net benefits for every pair of words. The authors carried out an experiment to evaluate their system in two languages, English and Spanish. In both cases, the results obtained after applying their algorithms to the output stems returned by the Porter stemmer and by a trigram approach (an n-gram corresponds to the *n* first letters of a word, then a trigram contains the first three letters of a word), with a window of 100 words, show a significant improvement with respect to the original stemmers in terms of performance. However, Larkey *et al.* (2002) apply the same technique to Arabic but their experiments yield better results with a simple stemmer, based on stripping predefined suffixes and prefixes and also removing stop-words, than with the same approach enriched with co-occurrence analysis.

Performance enhancement of the stemmer is not the only aspect that has been investigated in the literature. An example is the statistical stemmer proposed by Melucci and Orio (2003), where the most important contribution is that it requires no manual work, not even in the training phase, to generate the model containing the rules of the stemmer. This is possible because a hidden Markov model is used to determine for every term if a letter belongs to the stem or to the suffix. The parameters used by this hidden Markov model are calculated through an unsupervised training process using expectation maximization, where no morphological rules or list of known stems are needed. Regarding performance, the authors carried out an experiment where they compared the Porter stemmer with their system and determined that the performance was equivalent.

Other researchers have proposed stemmers that are not based on rules. Majumder *et al.* (2007) aim at creating stemmers for languages that lack linguistic resources (like morphological rules or a thesaurus), such as Bengali. The proposal consists of creating equivalent classes of words using an unsupervised, statistical and agglomerative hierarchical clustering algorithm based on a measure of distance between strings. Besides this, as in a hierarchical clustering algorithm, obtained classes can be merged depending on their similitude. After carrying out some experiments using as a baseline a non-stemming approach, two classical approaches (Porter and Lovins) and an adapted cluster-based approach using n-gram to cluster instead of a similarity measure (de Roeck and Al-Fares, 2000), their proposal has an equivalent performance both in English and French in comparison to the original Porter results, but in Bengali it increases considerably with respect to the non-stemming version (up to around 40% improvement both in recall and precision).

Another approach to develop stemmers for languages that are extremely complex in terms of morphology or that lack linguistic resources consists of using a cross-language stemmer, that is, using mechanisms, such as bilingual dictionaries or machine translators, to translate the documents from the source language to English and then applying to them any of the stemmers developed for English while associating the results to the source document (Popovic and Willett, 1992; Larkey *et al.*, 2002; Chen and Gey, 2002).

Conclusion

Stemming algorithms' purpose is quite simple and specific: find the morphological root of a word. However, no language strictly follows a deterministic set of rules, so it is difficult to achieve this purpose systematically. That is why a perfect stemmer, able to accurately obtain the stems of any term independently of its features, does not exist. Researchers have tried to provide some classifications and evaluation metrics to help users when selecting a stemmer according to the task they have to carry out and to the expected results. To achieve this, two metrics, recall and precision, have been adopted as performance measures. The problem of these two metrics is that they are highly dependent on other processes of the information retrieval pipeline. In fact, both are related to the accuracy of the ordered list of documents that best answer an information need posed through a query. Stemming only allows a reduction of related words to the same morphological root, and then to abstract the words in documents to concepts at a higher semantic level, but other algorithms in the information retrieval pipeline are in charge of translating these concepts into multidimensional vectors and of figuring out the similitude between every document and the query, based on a similitude metric that must also be defined. Thus, many factors can have an influence on the precision and recall of an information retrieval application. The problem is that almost no researcher exposes under which exact conditions and information retrieval algorithms and metrics they carry out their experiments to evaluate the performance of stemmers. Therefore, it becomes quite difficult to compare performance measures obtained from different experiments and researchers, as we cannot ensure that all the algorithms and metrics used, except for the stemming algorithm, work at least equivalently. The result is that there is not a good solution to evaluate a stemmer individually and independently of other information retrieval process factors, in terms of precision and recall.

To avoid possible interferences from these other processes in the information retrieval pipeline, some approaches have been proposed to evaluate stemmers' performance without the need to embed stemming into an information retrieval process, such as quantifying how much they compress the input vocabulary or characterizing which types of errors they generate. On the one hand, we have seen that stemmers produce two types of errors (over-stemming and under-stemming), which in turn define the strength of the stemmer. This strength is one of the main features that define a stemmer, as the rest of the properties and metrics are related to it. If a user needs to classify a collection of documents so that the number of groups is as low as possible and each group contains most of the topic-related documents, then a stronger stemmer is suitable as they provide a high recall and have a high conflation rate. An example could be a student that needs to explore exhaustively all the documents related with a topic within a collection of documents: even if some of the retrieved documents do not belong to the topic, the user can be sure that the vast majority of those which do deal with the topic will be present in the output. Oppositely, if the task requires the classification to return highly coherent groups (where most of their documents are topic-related), even if two or more of these groups are related in some way, then the precision offered by a weak stemmer is one of the main features to consider. Notwithstanding, no researcher has proposed an objective and absolute measure of a stemmer's strength. In fact, proposed classifications have been done in a relative way by ordering the stemmers according to the values obtained for each of them in terms of under-stemming and over-stemming (as strength directly depends on these two measures).

Besides these performance considerations, other constraints can have an influence on which stemmer is the best adapted to the needs of the user. For example, a strong stemmer can be the better option if the system has storage restrictions to handle the stemming process (e.g., smartphone), as these stemmers reduce considerably the dictionary associated to the documents that is needed to calculate the similitude measures between documents or between a document and a query. Even if no experiments have been explicitly proposed to evaluate the benefits or drawbacks of applying stemming or not in an information retrieval process, it is assumed that, as the dictionary of terms that is going to be manipulated by the subsequent algorithms in the information retrieval pipeline is considerably thinner, then their processing speed should also increase considerably. This can be a turning point in devices with hardware limitations (e.g., smartphones) or in remote systems where the exchanged dataflow should be as low as possible (e.g., client-server systems).

With respect to the usefulness of stemming terms in an information retrieval system, there is not wide agreement. Nevertheless, all experiments seem to demonstrate that the nature and the length of the collection's documents directly influence the results. As we have seen, short documents like abstracts are perfect candidates for stemming, as the co-occurrence rate of their terms is lower, and then conflating related words can make hidden thematic relations flourish. Furthermore, languages that are highly inflective benefit much more from stemming as their terms are morphologically more related, and most of the derivations or inflections applied to related words are defined by rules that allow a straightforward recognition of the affixes. Finally, both the nature of the input documents (and their vocabulary) and the purpose of the application (search, explore, classify, etc.) greatly conditions the usefulness of applying stemming in an information retrieval application.

Acknowledgements

This research work has been partially funded by the Spanish Ministry of Science and Innovation through research project TIN2009-14659-C03-02, and also by Universidad Politécnica de Madrid (UPM) through grant SBUPM-SIPMHXZ. The authors thank to copy-editors of the journal for their assistance in enabling them to satisfy the style requirements of the journal.

About the authors

Cristian Moral is a Ph.D. Candidate in the Escuela Técnica Superior de Ingenieros Informáticos at Universidad Politécnica de Madrid, Spain. He received his Master of Science in Computer Science both from Universidad Politécnica de Madrid and from Politecnico di Torino (Italy). Cristian is member of the Decoroso Crespo Laboratory since 2011, where he has participated in some R&D projects in the area of virtual environments. His current research interests are information retrieval, visualization and manipulation through adaptive virtual environments and human-computer interaction. He can be contacted at: cmoral@fi.upm.es.
Angélica de Antonio has been faculty member in the Escuela Técnica Superior de Ingenieros Informáticos at the Universidad Politécnica de Madrid since 1990. She received her Ph.D. in Computer Science in 1994. Angelica is Director of the Decoroso Crespo Laboratory since 1995, where she has led several R&D projects in the areas of intelligent tutoring systems, e-learning, virtual environments and intelligent agents. Her current research interests focus on virtual and augmented reality, adaptive systems and human-computer interaction. She can be contacted at: angelica@fi.upm.es.
Ricardo Imbert is an associate professor in the Escuela Técnica Superior de Ingenieros Informáticos at the Universidad Politécnica de Madrid since 2000. He received his Ph.D. in Computer Science in 2005. Ricardo is member of the Decoroso Crespo Laboratory since 1996, where he has led several R&D projects in the areas of intelligent tutoring systems, intelligent software agents, virtual environments and adaptive interactive systems. His current research interests deal with cognitive agent architectures, agent-based software engineering, human-computer interaction and interactive systems. He can be contacted at: rimbert@fi.upm.es.
Jaime Ramírez is an assistant professor in the Escuela Técnica Superior de Ingenieros Informáticos at Universidad Politécnica de Madrid, Spain. He received his Ph.D. in Computer Science in 2002. Jaime is member of the Decoroso Crespo Laboratory since 1996, where he has participated in several R&D projects in the areas of intelligent tutoring systems, e-learning, virtual environments and intelligent agents. His current research interests are intelligent tutoring systems, virtual environments for training, and user and student modelling using ontologies and data mining. He can be contacted at: jramirez@fi.upm.es.

References

- Adam, G., Asimakis, K., Bouras, C. & Poulopoulos, V. (2010). An efficient mechanism for stemming and tagging: the case of Greek language. In Rossitza Setchi, Ivan Jordanov, Robert J. Howlett and Lakhmi C. Jain , (Eds.), *Proceedings of the 14th International Conference on Knowledge-based and Intelligent Information and Engineering Systems*, **Part III**, 389-397. Berlin: Springer-Verlag.
- Aikhenvald, A.Y., Talmy, L., Bickel, B., Nichols, J., Corbett, G.G., Timberlake, A... *et al.*, Thompson, S. (2007). Language Typology and Syntactic Description. Volume 3: Grammatical Categories and the Lexicon. Shopen, T. (Ed.). Cambridge: Cambridge University Press
- Asian, J., Williams, H. E. & Tahaghoghi, S. M. (2005). Stemming Indonesian. *Proceedings of the 28th Australasian conference on Computer Science - ACSC '05*, **38**, 307-314. Newcastle, Australia: Australian Computer Society, Inc.
- Baeza-Yates, R. A. and Ribeiro-Neto, B. (1999). Modern information retrieval. Boston, MA: Addison-Wesley Longman Publishing Co., Inc.
- Bell, C. and Jones, K. P. (1979). Towards everyday language information retrieval systems via minicomputers. *Journal of the American Society for Information Science*, **30**(6), 334-339.
- Ben Abdessalem Karaa, W. (2013). A new stemmer to improve information. *International Journal of Network Security & Its Applications (IJNSA), 5*(4), 143-154
- Braschler, M. & Ripplinger, B. (2004). How effective is stemming and decompounding for German text retrieval? *Information Retrieval, 7*(3-4), 291-316.
- Chen, A. & Gey, F. (2002). Building an Arabic stemmer for information retrieval. *In Proceedings of TREC 2002*, 631-639. NIST, Gaithersburg.
- Chintala, D. R. & Reddy, E. M. (2013). An approach to enhance the CPI using Porter stemming algorithm. *International Journal of Advanced Research in Computer Science and Software Engineering, 3*(7), 1148-1156
- Çilden, E. K. (2010). Stemming Turkish words using Snowball. Retrieved 26 January 2014 from http://snowball.tartarus.org/algorithms/turkish/accompanying_paper.doc. (Archived by WebCite® at http://www.webcitation.org/6MvJeR7Wz)
- Comrie, B. (1989). *Language universals and linguistic typology: syntax and morphology*. Chicago, IL: University of Chicago Press
- Dawson, J. (1974). Suffix removal and word conflation. *ALLC Bulletin, 2*(3), 33-46
- de Roeck, A. N. & Al-Fares, W. (2000). A morphologically sensitive clustering algorithm for identifying Arabic roots. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, (pp. 199-206). Stroudsburg, PA: Association for Computational Linguistics.
- Dinçer, B.T. (2003). Stemming in agglutinative languages: a probabilistic stemmer for Turkish. In A. Yazıcı and C. Sener, (Eds.), *Computer and Information Sciences - ISCIS 2003*, (pp. 244-251.) Berlin: Springer
- Ekmekçioglu, F. Ç., Lynch, M. F. & Willett, P. (1996). Stemming and n-gram matching for term conflation in Turkish texts. *Information Research, 2*(2), paper 13. Retrieved from http://www.informationr.net/ir/2-2/paper13.html (Archived by WebCite® at http://www.webcitation.org/6NbaMhEBc)
- Eryiğit, G. C. & Adalı, E. (2004). An affix stripping morphological analyzer for Turkish. In M. H. Hamza, (Ed.), *Proceedings of the International Conference on Artificial Intelligence and Applications - AIA'04*, (pp. 299-304). Innsbruck, Austria: ACTA Press.

- Fernández, A., Díaz, J., Gutiérrez, Y. & Muñoz, R. (2011). An unsupervised method to improve Spanish stemmer. In *Proceedings of the 16th International Conference on Natural Language Processing and Information Systems*, (pp. 221-224). Berlin: Springer-Verlag.
- Frakes, W. B. (1984). Term conflation for information retrieval. *In Proceedings of the 7th annual international ACM SIGIR conference on Research and development in information retrieval*, (pp. 383-389). Swinton, UK: British Computer Society.
- Frakes, W.B. & Fox, C.J. (2003). Strength and similarity of affix removal stemming algorithms. *SIGIR Forum, 37*(1), 26-30.
- García Figuerola, L.C., Gómez-Díaz, R., Zazo Rodríguez, Á. F. & Alonso Berrocal, J. L. (2001). Stemming in Spanish: a first approach to its impact on information retrieval. *Results of the CLEF 2001 Cross-Language System Evaluation Campaign. Working Notes for the CLEF 2001 Workshop*, (pp. 197-202). Springer-Verlag
- Hankamer, J. (1984). *Turkish generative morphology and morphological parsing.* Paper presented at the *Second International Conference on Turkish Linguistics*, Istanbul, Turkey
- Harman, D. (1991). How effective is suffixing? *Journal of the American Society for Information Science, 42*(1), 7-15.
- Hull, D. A. (1996). Stemming algorithms - a case study for detailed evaluation. *Journal of the American Society for Information Science, 47*, 70-84
- Kent, A., Berry, M. M., Luehrs, F. U. & Perry, J. W. (1955). Machine literature searching VIII. Operational criteria for designing information retrieval systems. *American Documentation, 6*(2), 93-101
- Kettunen, K. (2009). Reductive and generative approaches to management of morphological variation of keywords in monolingual information retrieval: an overview. *Journal of Documentation, 65*(2), 267-290
- Kettunen, K., Kunttu, T. & Järvelin, K. (2005). To stem or lemmatize a highly inflectional language in a probabilistic IR environment? *Journal of Documentation, 61*(4), 476-496
- Korenius, T., Laurikkala, J., Järvelin, K. & Juhola, M. (2004). Stemming and lemmatization in the clustering of Finnish text documents. In *Proceedings of the thirteenth ACM International Conference on Information and Knowledge Management*, (pp. 625-633). New York, NY: ACM Press.
- Kraaij, W. & Pohlmann, R. (1994). Porter's stemming algorithm for Dutch. In L.G.M. Noordman & W.A.M. de Vroomen (Eds.), *Informatiewetenschap 1994: Wetenschappelijke bijdragen aan de derde STINFON Conferentie*, (pp. 167-180). Leiden, Netherlands: Stichting Informatiewetenschap Nederland
- Kraaij, W. & Pohlmann, R. (1996). Viewing stemming as recall enhancement. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, (pp. 40-48.) New York, NY: ACM Press.
- Krovetz, R. (1993). Viewing morphology as an inference process. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 191-202. New York, NY: ACM Press.
- Larkey, L.S., Ballesteros, L. & Connell, M. E. (2002). Improving stemming for Arabic information retrieval: light stemming and co-occurrence analysis. In *Proceedings of the 25th annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, (pp. 275-282). New York, NY: ACM Press.
- Lennon, M., Pierce, D. S., Tarry, B. D. & Willett, P. (1988). Document retrieval systems. In P. Willett (Ed.), *Document retrieval systems*, (pp. 99-105). London: Taylor Graham Publishing.
- Lovins, J. B. (1968). Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics, 11*(1/2), 22-31. Retrieved 26 January 2014 from http://mt-

archive.info/MT-1968-Lovins.pdf. (Archived by WebCite® at http://www.webcitation.org/6MmQ25qVU)

- Luhn, H. P. (1957). A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of Research and Development, 1*(4), 309-317.
- Majumder, P., Mitra, M. & Pal, D. (2008). Bulgarian, Hungarian and Czech stemming using YASS. In Carol Peters, Valentin Jijkoun, Thomas Mandl, Henning Müller, Douglas W. Oard, Anselmo Peñas, Vivien Petras & Diana Santos (Eds.), *Advances in multilingual and multimodal information retrieval: 8th Workshop of the Cross-Language Evaluation Forum, CLEF 2007, Budpest, Hungary, September 2007. Revised selected papers*, (pp. 49-56). Berlin: Springer-Verlag. (Lecture Notes in Computer Science, 5152).
- Majumder, P., Mitra, M., Parui, S. K., Kole, G., Mitra, P. & Datta, K. (2007). YASS: yet another suffix stripper. *ACM Transactions on Information Systems, 25*(4), paper 18. Retireved from http://cse.iitkgp.ac.in/~pabrita/paper/stemmer.pdf (Archived by WebCite® at http://www.webcitation.org/6NbketMkE)
- Manning, C. D., Raghavan, P. & Schütze, H. (2008). *Introduction to information retrieval*. New York, NY: Cambridge University Press
- Mayfield, J. & McNamee, P. (2003). Single n-gram stemming. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, (pp. 415-416). New York, NY: ACM Press.
- Melucci, M. & Orio, N. (2003). A novel method for stemmer generation based on hidden Markov models. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, (pp. 131-138). New York, NY: ACM Press.
- Meyer, C. F. (2010). *Introducing English linguistics*. Cambridge: Cambridge University Press
- Miller, G. A. (1995). WordNet: a lexical database for English. *Communications of the ACM, 38*(11), 39-41.
- Moulinier, I., McCulloh, J. A. & Lund, E. (2001). West Group at CLEF 2000: Non-English monolingual retrieval. In *Revised Papers from the Workshop of Cross-Language Evaluation Forum on Cross-Language Information Retrieval and Evaluation*, (pp. 253-260). London: Springer-Verlag.
- Orengo, V. & Huyck, C. (2001). A stemming algorithm for the Portuguese language. In *Proceedings of the Eighth International Symposium on String Processing and Information Retrieval 2001 (SPIRE 2001)*, (pp. 186-193). Laguna de San Rafael, Chile: IEEE Computer Society Press.
- Otxandorena, M. (2010). *A Basque stemming algorithm*. Retrieved 26 January 2014 from http://snowball.tartarus.org/algorithms/basque/stemmer.html. (Archived by WebCite® at http://www.webcitation.org/6MmQC5KbS)
- Paice, C. D. (1990). Another stemmer. *SIGIR Forum, 24*(3), 56-61.
- Paice, C. D. (1994). An evaluation method for stemming algorithms. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, (pp. 42-50). New York, NY: Springer-Verlag.
- Patil, C. G. & Patil, S. S. (2013). Use of Porter stemming algorithm and SVM for emotion extraction from news headlines. *International Journal of Electronics, Communication & Soft Computing Science and Engineering, 2*(7), 9-13
- Peng, F., Ahmed, N., Li, X. & Lu, Y. (2007). Context sensitive stemming for Web search. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, (pp. 639-646). New York, NY: ACM Press.
- Pirkola, A. (2001). Morphological typology of languages for IR. *Journal of Documentation, 57*(3), 330-348

- Popovic, M. & Willett, P. (1992). The effectiveness of stemming for natural-language access to Slovene textual data. *Journal of the American Society for Information Science, 43*(5), 384-390.
- Porter, M. F. (1980). An algorithm for suffix stripping. *Program: Electronic Library and Information Systems, 40*(3), 211-218.
- Porter, M. F. (2001). Snowball: a language for stemming algorithms. Retrieved 26 January 2014 from http://snowball.tartarus.org/texts/introduction.html. (Archived by WebCite® at http://www.webcitation.org/6MmQJZdhV)
- Salton, G. (1971). *The SMART retrieval system - experiments in automatic document processing*. Upper Saddle River, NJ: Prentice-Hall, Inc.
- Salton, G. (1989). *Automatic text processing: the transformation, analysis, and retrieval of information by computer*. Boston, MA: Addison-Wesley Longman Publishing Co., Inc.
- Smirnov, I. (2008). *Overview of stemming algorithms*. Retrieved 26 January 2014 from http://the-smirnovs.org/info/stemming.pdf. (Archived by WebCite® at http://www.webcitation.org/6MmQO7y0Sv)
- Taghva, K., Elkhoury, R. & Coombs, J. (2005). Arabic stemming without a root dictionary. In *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05), Volume I*, (pp. 152-157). Washington, DC: IEEE Computer Society.
- Van Rijsbergen, C. J. (1979). *Information Retrieval*. (2nd. ed.). Newton, MA: Butterworth-Heinemann
- Voorhees, E. M. (1994). Query expansion using lexical-semantic relations. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, (pp. 61-69). New York, NY: Springer-Verlag.
- Whaley, L. J. (1997). *Introduction to typology: the unity and diversity of language*. Thousand Oaks, CA: Sage Publications.
- Xerox Corporation Ltd. (1994). *Xerox linguistic database reference* (English version 1.1.4). Norwalk, CT: Xerox Corporation Ltd.
- Xu, J. & Croft, W.B. (1998). Corpus-based stemming using cooccurrence of word variants. *ACM Transactions on Information Systems, 16*(1), 61-81.

How to cite this paper

Moral, C., de Antonio, A., Imbert, R. & Ramírez, J. (2014). A survey of stemming algorithms in information retrieval/ *Information Research*, **19**(1) paper 605. [Available at http://InformationR.net/ir/19-1/paper605.html]