Artif Intell Rev (2017) 48:157–217
DOI 10.1007/s10462-016-9498-2

A systematic review of text stemming techniques

Jasmeet Singh[1] · **Vishal Gupta**[1]

Published online: 1 August 2016

Abstract Stemming is a program that matches the morphological variants of the word to its root word. Stemming is extensively used as a pre-processing tool in the field of natural language processing, information retrieval, and language modeling. Though a lot of advancements have been made in the field, yet organized arrangement of the previous work and efforts are lacking in this field. In this paper, we present a review of the text stemming theory, algorithms, and applications. It first describes the existing literature relevant to text stemming by classifying it according to certain key parameters; then it describes the deep analysis of some well-known stemming algorithms on standard data sets. In the end, the current state-of-the-art and certain open issues related to unsupervised stemming are presented. The main aim of this paper is to provide an extensive and useful understanding of the important aspects of text stemming. The open issues and analysis of the current stemming techniques will help the researchers to think of new lines to conduct research in future.

Keywords Stemming · Natural language processing · Information retrieval · Language modeling

1 Introduction

Today in the age of Internet, the presence of different communities of the society on the World Wide Web has increased remarkably. A large amount of data is available in digital form in multiple languages. The major concern, today, is to identify the challenges for the efficient tackling of document bases in a large number of languages and to develop additional technology to prepare for those challenges. Hence, development of intelligent tools for language processing and information retrieval has become an active research area. Earlier the language processing tools were mainly developed for English. But, with an increase in

✉ Vishal Gupta
vishal_gupta100@yahoo.co.in

[1] University Institute of Engineering and Technology, Panjab University, Chandigarh, India

multi-lingual documents and insufficient tools and resources of languages other than English research work is now also being done for non-English languages. The various areas of information retrieval and natural language processing require certain text pre-processing tools for lexical, morphological, syntactic and semantic level analysis. Stemming is one of the numerous pre-processing tools and is useful in the areas of information retrieval and natural language processing such as text classification, clustering, searching, summarization, POS tagging, etc.

1.1 Stemming

The basic word form in most of the languages is modified to form variant word forms according to the function of the word in a sentence. These word forms are formed through different linguistic processes such as compounding (combination of two or more words), affixation (addition of prefixes and/or suffixes), conversion (formation of new words from existing ones), etc. These word forms often share the same meaning. Stemming is a procedure in which the various morphological variants of words are matched to their stem (root, base word). For instance, the words *maintaining, maintained, maintenance* are matched to their root word *maintain* through stemming. The programs that perform stemming are called stemmers. Stemming is the simplest type of language processing used in IR system and is found to be more beneficial in languages with complex morphology where a single word has a large number of variants (Xu and Croft 1998). Though text stemming started in the late sixties (Lovins 1968), it has experienced tremendous growth in recent years, and a large number of techniques and algorithms have been proposed to handle this task. However, to design completely unsupervised language independent stemmers is a great challenge.

Stemming has been perceived in different perspectives and researchers in the field of information retrieval and linguistic processing consider it to be desirable and an important step for different reasons (Manning et al. 2008). Firstly, stemming is viewed as a tool to improve retrieval accuracy. Stemming reduces the variant forms to the root word during indexing and searching of documents. The problem of vocabulary mismatch between the documents and the queries is addressed, and all those documents that do not exactly match the query terms are also retrieved. In Information retrieval systems, stemming is considered as a recall enhancing device, but for languages with complex morphology, it improves precision as well by promoting the relevant documents in superior ranks (Xu and Croft 1998).

Secondly, stemming is viewed as a mechanism to reduce the size of the index file since the various variant terms are reduced to a single term (Melucci and Orio 2003; Bhamidipati and Pal 2007). Sometimes stemming is found to be so useful, that the size of the index file is reduced to half of the original size. From another perspective, stemming is also viewed as clustering or feature reduction/selection mechanism where the major aim is to select the most appropriate dimension or class or a concept. Stemming is also viewed as a mechanism to normalize the different concepts or senses used in the query terms (Krovetz 1993). The stemming rules help in identifying which word forms are related to each other and such relations can be used by the stemmer to resolve the meaning of the word. For instance, suffixes are only added to the stems with specific parts-of-speech, this knowledge can be used to differentiate between two homographs [*intimation* derived from *intimate* (verb) and *intimately* derived from *intimate* (adjective)]. Hence, stemming is a mechanism which is widely accepted in terms of consistency and acceptance by the users.

1.1.1 Stemming and lemmatizing

Stemming and lemmatizing are often considered as sibling processes and put under the same roof. Both processes are related and perform a similar function of reducing the variant words in the input text. The basic difference between both the processes lies in their outputs. The output product of stemming is 'stem' and that of lemmatization is 'lemma'. Stems usually have distinct meaning and are often task-oriented. Stem is a part of the word (with or without meaning) which are used to form new words through various linguistic methods such as compounding (e.g. six-pack, day-dream) or affixation (e.g. perish-able, dur-able) (Huddleston 1988). The stem may be valid, fully understandable word (free stem) or invalid word which requires an affix to make a word (bound stem). For instance, 'perish' is a free stem and 'dur' is a bound stem. Lemmas, on the other hand, are valid linguistic components and a dictionary form of a lexeme. Lexeme corresponds to a collection of all the word variant forms that have similar meaning and lemma is one particular variant used to represent the lexeme. For instance, *run*, *ran*, *runs, running* are different forms of lexeme which are represented by the lemma 'run'.

Stemming is a simpler, easier and faster process that makes use of rules to determine the stem without considering the vocabulary, context of the word or part-of-speech whereas lemmatization is a comparatively complex procedure which first determines the part-of-speech and context of the word to return the lemma (Jivani 2011). Lemmatization performs complete morphological analysis of the words to determine the lemma whereas stemming removes the variations which may or may not be morphologically correct word forms. For instance, the word forms, *introduces, introducing, introduction* are mapped to lemma 'introduce' through lemmatizer, but a stemmer will map it to the stem 'introduc'. This oddity cannot be considered as a flaw of stemmers, as the document keywords and queries are invisibly stemmed for a user.

In some cases, stemmers and lemmatizers can replace each other as stemmers cannot be used where the desired output should be a valid word of a language. On the other hand, stemmers can be designed to remove derivational suffixes whereas lemmatizers only remove the inflectional variations (Brychcín and Konopík 2015). Stemmers are semantically oriented and have the tendency to combine lexemes that are semantically related. Moreover, stemmers can be developed in an unsupervised manner without the use of any linguistic resource or expert, but currently, no method has been proposed in the literature for training a lemmatizer in an unsupervised manner.

1.2 Motivation for conducting the survey

The motivation for conducting this survey is to highlight the present status of stemming by finding its historical developments. An extensive survey of the stemming techniques is conducted and the various methods are compared using various metrics. The following facts specifically motivate this survey:

- To present a comprehensive review of various stemming techniques, covering not only the functioning details of the methods but also identifying their distinguishing features. A number of useful tables highlighting features, advantages, disadvantages and performance analysis of various stemming techniques are synthesized cohesively.
- To analyze the retrieval performance of various well-known stemming techniques in different language families. The retrieval performance of stemmers is also evaluated on various factors such as the size of training data, nature of documents, type of training

and different indexing and searching schemes. Besides, information retrieval, stemmer performance is compared in web searching and text classification tasks.
– To identify challenges and future research directions for unsupervised stemming. We highlighted various open issues related to unsupervised statistical stemming such as discovering rules other than affix stripping, learning advanced semantic relations from the corpus, unsupervised parameter tuning, evaluation independent of IR system, etc.

The rest of the article is organized as follows. In Sect. 2, classification of various stemming techniques is presented with reference to previous work. Section 3, reviews various language specific stemming techniques proposed in the literature. Besides English, linguistic stemmers in nearly thirty languages belonging to nine language families are reviewed in this section. In Sect. 4, methods related to unsupervised statistical stemming are described. Various evaluation mechanisms used for evaluating stemmers are discussed in Sect. 5. The evaluation results and analysis of some well-known stemmers on standard TREC, CLEF, and FIRE collections is also presented in this section. In Sect. 6, the performance of stemmers according to various factors and tasks is analyzed. Applications of stemming in various fields are discussed in Sect. 7. Various open issues and future directions related to unsupervised stemming are presented in Sect. 8. Section 9 concludes this article.

2 Classification of stemming techniques

Stemming is a well-studied technology and a number of stemmers based on different techniques and flavors are presented in the literature. The classification of stemming techniques is presented in Fig. 1. Broadly stemming techniques are classified into two categories: *Language Specific* (Rule-Based) and *Statistical* (Corpus-Based) techniques.

Language Specific or *Rule-based* stemmers make use of certain pre-defined language-related rules to map the morphological variants of the word to its base form. These language-related rules are created manually by the language experts or linguists. The quality of the output of rule-based stemmers is quite better than statistical stemmers because they not only strip the affixes from the word but can also change the complete word ('ate' to 'eat'). The creation of rule-based stemmers is very time-consuming, and moreover, it requires linguistic experts and resources such as dictionaries, stem tables, etc. Language specific stemming methods are further classified into three categories: *Table Lookup*, *Affix Stripping*, and *Morphological*.

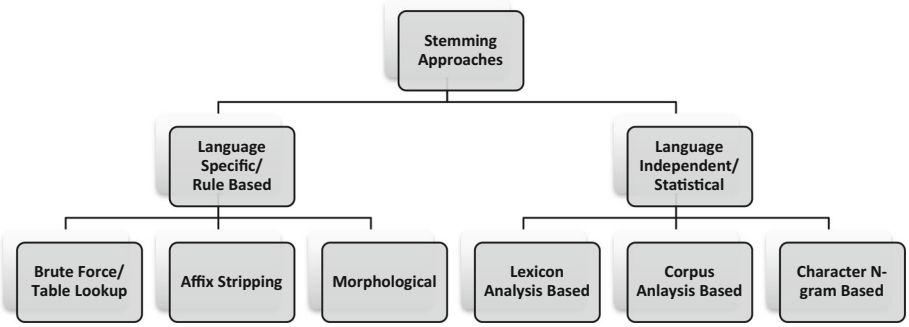

Fig. 1 Classification of stemming techniques

- *Table lookup (brute force stemming)* These techniques make use of a lookup table that contains the root word corresponding to the inflected or derived words (Frakes 1992). In order to find the root of the word, the table is checked. If the match is found, then the root is returned. These algorithms are also called as dictionary based algorithms. These are simple and easy-to-use techniques which can take care of the exceptional cases as well. But these techniques require various language resources, and they cannot handle the words outside the dictionary.
- *Affix stripping algorithms* The prefix or suffix of the word is called affix. Affix removal algorithms delete suffix and/or prefix of the word according to specific rules or suffix list (Baeza-Yates and Ribeiro-Neto 2011; Frakes 1992). A lot of work has been done in suffix stripping in comparison to prefixes. Development of rules for the stemmer requires the complete expertise of the language and various language resources. These techniques cannot handle variations caused due to compounding, spelling variations and produce a number of errors as the words produced after stripping of affixes are sometimes not real words.
- *Morphological stemmers* These stemmers take into account the morphology of the language while stemming. Inflectional stemmers consider the inflectional morphology i.e. they can detect the changes in the word caused due to the syntax such as forms of nouns, verbs, changing singular into plural but the part-of-speech (POS) remains same (Krovetz 1993). Derivational stemmers take into account the derivational morphology i.e. change in the category such as nominalization (a noun phrase generated from some another class such as 'informer' from 'inform'), deadjectival (word derived from adjectives such as 'happiness' from 'happy'), deverbal (word derived from verb which is usually noun or adjective such as 'readable' from 'read'), and denomial (word derived from noun such as 'useful' from 'use'). These stemmers take into account the dictionary information such as context, word meanings, vocabulary, etc. The efficiency of these stemmers is quite high as these algorithms consider both the syntax as well as the semantics of the language. But the development of these algorithms requires complete knowledge of the language and its morphology.

Statistical techniques are based on unsupervised learning of the language by analyzing the lexicon or finding the co-occurrence or context of the words in the corpus. These are also called corpus-based techniques. These algorithms also perform suffix stripping but after performing some statistical analysis on the corpus (Paik et al. 2011a). Statistical techniques incorporate new language into the system with very little efforts, and this is useful especially for applications related to information retrieval. Moreover, these techniques can deal with languages that have complex morphology and sparse data. The major advantage of statistical techniques is that it does not require any prior knowledge of the language or language resources which are useful for many languages where the resources are either not available or are incomplete to provide effective results. Statistical techniques involve a lot of computations and take time as they learn the morphology of the language from the corpus on their own. Statistical techniques can be further divided into following categories:

- *Lexicon analysis based* These stemmers understand the morphology of the languages by analyzing the lexicon of the language (Paik and Parui 2011). Word variants are identified from the lexicon using different methods such as computation of frequencies of substrings (Goldsmith 2001; Bacchin et al. 2005), string distances or similarities (Majumder et al. 2007b)
- *Corpus analysis based* These stemmers use the context or co-occurrences statistics of the corpus words to perform stemming. The various statistical analysis methods such as

Table 1 Comparison of stemming techniques

Features	Language specific			Statistical
	Table lookup	Affix stripping	Morphological	(Lexicon, corpus, N-gram)
Use of suffix list/rule table	No	Yes	Yes	No
Use of corpus	No	No	No	Yes
Use of stem table/ context/dictionary	Yes	No	Yes	No
Language expertise and dependence	Yes	Yes	Yes	No
Computations	No	No	No	Yes
Execution time	Fast	Medium	Slow	Fast

expected maximization (Xu and Croft 1998), co-occurrence strength (Paik et al. 2011b, 2013), distributional similarity (Bhamidipati and Pal 2007), etc. are employed to learn the morphological rules for stemming.

– *Character N-gram based* These stemmers use n-grams derived from the words of the language to perform stemming. The frequency or probability information derived from n-grams can help in identifying the variants as the frequency or probability of prefixes and suffixes is large as compared to roots or stems (Smirnov 2008).

Besides techniques mentioned above, stemmers are also developed through a hybrid of two or more techniques wherein stemming is performed by combining two or more than two algorithms. For instance, table lookup and suffix stripping approach can be combined; where the word to be stemmed is first searched in the lookup table. If the stem is found then it is returned, otherwise suffix stripping rules are applied to the word. Table 1 compares the various stemming approaches on the basis of some distinguishing features.

Stemmers can also be classified as *light* or *aggressive* stemmers. Light stemmers tend to under stem the words. In doubtful cases, light stemmers leave the word intact rather than creating too short stems (for instance, reducing the word *jumping* to *jumpi*). Thus, light stemmers favor precision over recall (Brychcín and Konopík 2015). Aggressive stemmers work another way around and decrease precision. They often tend to overstem the words and perform stemming even at the risk of forming too short stems. For instance, *generate* and *general* both are reduced to *gene*.

3 Language specific stemmers

A variety of language-specific stemmers are developed that perform the task of stemming using language specific rules. The type of algorithm and the nature of the work involved depend on a number of factors such as whether suffix list is used or not, rule table for suffix stripping is involved or not, dictionary being used and above all the purpose or need for performing stemming. Initially, most of the work in this field has been done for the English language but with the rapid increase of multi-lingual documents, stemmers other than the English language have been developed to tackle the challenges in this field. Various linguistic

stemmers for the English language proposed in the literature are described in Sect. 3.1, and stemmers for languages other than English are described in Sect. 3.2.

3.1 Linguistic stemmers in English

The performance of stemming algorithms for English in Information Retrieval has produced varied results and is always a debated affair. Harman (1991) tested a few stemming algorithms on English dataset and reported a little effect on the overall performance. The successive studies (Krovetz 1993; Hull 1996; Xu and Croft 1998) reported a positive effect of stemming on ad hoc retrieval tasks for the English language which has a simple morphology. Some of the well-known classical English stemmers are discussed below:

- *Lovins stemmer* Lovins (1968) was the first stemming algorithm published in the literature. The proposed technique is fast and very simple to use as it performs stemming in two simple steps: removal of endings based on longest match principle and recoding of the stem. Lovin classified the 294 endings into 11 classes depending upon the length of the suffix which varies from 2 characters to 11 characters. Each ending is linked to one of the 29 context-sensitive conditions. It first makes use of lookup table to remove endings and apply one of the context-sensitive rules. In the next step, the stems are recoded using one of the 35 transformation rules. The major aim of this step is to deal with multiple spellings of the word (analyses: analyzes), handling double consonants (submitted: submitt: submit) and irregular variations (matrix: matrices). Finally, the stems are conflated using the partial-matching method in which the words are grouped if their stems are quite close but not compulsory same. The recoding phase can increase conflation rate, but it can increase errors also by conflating unrelated words together which mostly share the same stem (Moral et al. 2014). For example, the unrelated words *probate* and *probe* are stemmed to stems *prob* and *probe* and are then grouped together by the partial-matching procedure. Lovins algorithm is fast, but it missed many suffixes as the author used only technical vocabulary.
- *Dawson stemmer* Dawson (1974) proposed an extended and improved version of Lovins stemmer. It is fast single pass suffix removal context-sensitive stemmer. It is also based on longest match principle, but it has a more exhaustive list of nearly 1200 endings, although there is no record of the list in the literature. This stemmer filtered the ending list of Lovins stemmer by first adding the plurals and different combinations of the endings, thereby making a total of 500 and then adding more variations and flexions of suffixes thus making a total of 1200. The list of suffixes are stored in reverse order and are indexed by the total length of the suffix and then by the last letter thereby making the access rapid. The basic difference between Lovins and Dawson stemmer is that there are no recoding rules to handle spelling exceptions in Dawson stemmer as it was not found to be reliable in Lovins stemmer. Dawson stemmer makes use of a partial matching procedure which matches the stems that are similar in some limits. For example, the words *explanation* and *explain* are stemmed to *explan* and *explain* by suffix stripping process and then are conflated together using the partial matching procedure. The output of suffix stripping from the stemmer is given to IR system, and it is entirely the responsibility of IR system to perform the partial matching procedure by grouping the words whose stems are close (similar to some limits), but not necessarily same.
- *Porter stemmer* Although Lovins stemmer is the first published stemmer, yet Porter stemmer (1980) is the most popular and widely used stemmer in IR tasks possibly because of the balance between efficiency and simplicity. It is based on the fact that the various

suffixes in the English language are comprised of simple and smaller suffixes. Porter Stemmer is a non-recursive rule-based stemmer which makes use of nearly 60 rules that are applied successively in five steps. In each step if the suffix and the condition match in the word, then the suffix in the word is removed as per the rule. Porter suggested that any word or even part of a word can be expressed as (C) $(VC)^m$ (V); where C denotes one or more consonants, V denotes one or more vowels; m denotes a measure of a word or word part. The rules in Porter stemmer are expressed as (Condition) E1 $\rightarrow$ E2. As an example, according to the rule (m $>$ 0) FUL$\rightarrow$ ϕ the word HOPEFUL would be mapped to its root word HOPE. Porter stemmer is efficient in terms of complexity and readability, but the errors like over-stemming (probe/probable) are well known.

Being dissatisfied with the problems of misinterpretation of the algorithm, changes made by programmers for improvements and errors in encoding, the author circulated the implementation of the algorithm in C and Java. Moreover, he developed a standard framework for the development of stemmers named Snowball (Porter 2001). Snowball is a framework in which the stemming rules can be written in natural language, and the compiler converts the rules into ANSI C or Java code. The framework is available at http://snowball.tartarus.org. Snowball is a useful and an important resource in designing stemmers in a number of languages and currently implementations in languages like Romanic, Germanic, Scandinavian, Russian, Finnish, etc. are available on the framework website.

- *Paice/Husk stemmer* Paice/Husk (1990) suggested an iterative stemming algorithm wherein only one rule table is used. Each rule in the rule table performs either removal of suffix or substitution of the suffix with another suffix. So it does not require any separate recoding stage as in the case of other stemmers. The rules in the rule table are grouped together into various groups according to the last letter of the ending, also called as section letter. The stemming procedure is fast as the rule table is rapidly accessed by just matching the last letter of the word. Paice/Husk pointed that for each group in the rule table, the sequence of rules is important and moreover there are certain rules that are limited to intact words. Each rule in Paice/Husk stemmer is composed of five different parts: (i) A string in reverse order corresponding to the ending; (ii) "*" for the intact words (optional); (iii) number of characters to be removed from ending; (iv) a string that is to be appended after removing the characters from the ending (optional); (v) ">" is used as a symbol for continuation and "." is used as symbol for termination. As an example, the rule "ylp0." denotes that if the ending is *ply,* then do not change it and terminate.
- *Krovetz stemmer (KSTEM)* Krovetz (1993) pointed out the limitations of affix stripping stemmers that these stemmers produce the stem of the word, but the results are improper as they do not make use of dictionary or lexicon. The author suggested that if the documents are indexed by meanings of the words rather than words, then the performance of stemmer would be better. KSTEM is an inflectional and derivational stemmer whose stems are words rather than truncated forms. The inflectional stemming algorithm works in three steps: firstly, the singular form corresponding to plurals are obtained; secondly, the present tense form corresponding to past tense is obtained and lastly, -ing is deleted, and double letters at the end are converted into single letters. After removal of suffixes, recoding of the resultant word is done using a support dictionary to obtain meaningful words. The stemmer handles only three basic inflections, so it has weak strength and is used in conjunction with other stemmers (Jivani 2011).
- *XEROX stemmer* Hull (1996) proposed an inflectional and derivational stemmer that is based on Xerox linguist, which has formulated lexical systems for some languages including English. The inflectional system can map various word forms to root words

Table 2 Features of classical english stemmers

Stemmer	Number of suffixes	Number of rules	Recoding phase	Partial matching	Context sensitive rules
Lovins	294	29	Yes	Yes	Yes
Dawson	1200	No	No	Yes	Yes
Porter	51	60	No	No	Yes
Paice/Husk	No	115	No	No	Yes
Krovetz	5	Unknown	Yes	No	No
Xerox	Unknown	Unknown	No	No	Yes

that can be found in the lexicon. Four rules are used for inflectional mapping: Plural nouns are mapped to the singular; the verb is mapped to its infinitive form (dancing to dance); adjectives are being mapped to their positive form (cleaner/cleanest to clean), and the pronouns are mapped to their nominative form (his to he).

The Xerox derivational system maps the various word forms to the root forms that are related in terms of context and meanings and are present in the lexicon. The system is based on the removal of affixes i.e. both suffix and the prefix and considers certain irregular word forms also. Some suffixes used by the derivational stemmer are -ness, -ate, -ish, -ive, -less, -ure etc. and some prefixes used are -anti, -dis, -di, -super, etc.

Table 2 sums up the features of the classical English stemmers presented above. This summary explains the strengths and scope of the benchmark stemmers. A number of new stemmers for English are proposed based on these classical stemmers so as to address the limitations of these stemmers.

3.1.1 Distinguishing features of rule-based stemmers

Various rule-based stemmers discussed above make use of linguistic features, but they perform differently and report different experimental results. The major reason for these differences is that these rule-based stemmers use different rules, different suffix lists, different dictionaries, etc. In this subsection, we attempt to identify the major differences in the methods belonging to this category.

Dawson (1974) stemmer considered the idea proposed by Lovins (1968) and attempted to improve the rule sets and rectify the basic existing errors. Both the stemmers are developed on the same lines and make use of context sensitive suffix removal lists. The context-sensitive conditions in both stemmers are similar as they impose minimum length restriction on the resultant stem or restriction on removal of suffixes in the presence of certain set characters in the resultant stem. The major difference between the Lovins and Dawson stemmers lies in the methods used to handle the spelling variation problems. Lovins stemmer uses recoding procedure as part of the main stemming algorithm to solve the issue of spelling exceptions. In this procedure, some transformation rules are applied to recode the stem. Dawson stemmer, on the other hand, employs a partial matching scheme which tries to match the stems that are similar in some limits. This partial matching procedure is not part of the stemming algorithm and is entirely the responsibility of the application system. The author warns that stemmer performance would be degraded a lot without this partial matching procedure. Another difference between the two stemmers is that the Lovin stemmer uses an incomplete

Table 3 Performance of classical suffix stripping stemming techniques

Parameters	Stemmers performance
Under-stemming rate	Porter > Lovins > Paice/Husk
Over-stemming rate	Paice/Husk > Lovins > Porter
Strength	Paice/Husk > Lovins > Porter
Compression factor	Paice/Husk > Lovins > Porter
Storage needs	Porter > Lovins > Paice/Husk
Recall	Paice/Husk > Lovins > Porter
Precision	Porter > Lovins > Paice/Husk

list of endings containing only 294 endings which have been elaborated by Dawson by adding plurals and other variations of suffixes thereby making a total of 1200. Dawson stemmer is not as popular as Lovins stemmer because of its complexity and unavailability of standard implementation.

Porter (1980) and Paice/Husk (1990) stemmers, on the other hand, use a set of context-sensitive rules to perform stemming. Paice/Husk stemmer is comparatively heavier than Porter stemmer and performs aggressive stemming. The major differences between the two stemmers is that the Porter stemmer employs non-recursive algorithmic rules in which the word to be stemmed goes through series of five steps whereas Paice/Husk stemmer employs rule execution approach where the rules are being stored in an external file, and the suffixes are removed from the word in an indefinite number of steps. The Porter stemmer is based on the assumption that various suffixes are formed by combining simpler and smaller suffixes and the set of rules in the stemming method are concise enough to remove these complex suffixes by clearing simple and small suffixes at each step. The algorithm, therefore, avoids the need for any separate partial matching or recoding stage for spelling exceptions. Paice/Husk stemmer is an iterative method wherein a selected rule deletes or replaces the ending. The ability to append new characters and to delete some characters is equivalent to recoding phase thereby solving the problem of spelling exceptions.

A number of experiments (Lennon et al. 1988; Harman 1991; Paice 1994; Hull 1996) have been performed to evaluate the performance of three widely used suffix stripping methods namely Lovins, Porter and Paice/Husk. These stemmers have been compared on the parameters such as over-stemming errors (distinct words conflated together), under-stemming errors (words with same stems are not conflated together), strength (aggressiveness with which stemmers clear terminations), compression factor (reduction in size of index), storage needs, precision (fraction of relevant documents to total documents retrieved by an IR system) and recall (fraction of relevant documents retrieved to total relevant documents). All these parameters are described in detail in Sect. 5.1. Table 3 sums up the performance of these stemmers on the basis of various parameters.

3.2 Linguistic stemmers in other languages

A number of stemmers in languages other than English are proposed in literature either by modifying the classical approaches or by developing new approaches to tackle the problems of the languages. Although stemming in English showed mixed results but languages with complex morphology where a single stem has a large number of variants like Romanic languages (Krovetz 1993; Popovic and Willet 1992), Dutch (Kraaij and Pohlman 1994), Finnish, Hungarian (Majumder et al. 2008; Savoy 2006), Greek (Adam et al. 2010), etc. stemming is found to be quite useful. For languages rich in morphology, stemming not

only increases recall but also increases precision by promoting the relevant documents in superior ranks. A number of studies (Popovic and Willet 1992; Kraaij and Pohlman 1994; Chen and Gey 2002) proved that the efficiency of the stemmer increases with increase in the morphological complexity of the language. In our study, we found the application of stemming algorithms in nearly 30 languages grouped into nine language families (Romanic, Germanic, Balto-Slavic, Semitic, Iranian, Malayo-Polenisyan, Uralic, Turkish and Hellenic).

– *Stemming in Romanic languages* Romanic languages are rich in morphology in which the morphological variation of root words is quite intense. In this category, most of the stemming work has been reported for Spanish, French, Italian and Portuguese language. Among Spanish stemmers, Honrado et al. (2000) proposed a popular stemming algorithm and pointed out that a simple Porter style stemmer cannot be developed for the Spanish language because of its greater morphological complexity. The proposed technique combines rule-based and table lookup approaches in which nearly 300 stemming rules and two dictionaries namely *I-Dictionary* (containing 5500 entries) and *RiV* Dictionary (containing 4000 entries) are used. Another rule-based Spanish stemmer is proposed by Figuerola et al. (2001) that makes use of Finite State Machine (FSM) to implement language-specific rules and lexicon. The authors also studied the effect of inflectional, derivational stemmer on the Spanish CLEF document set and reported that inflectional stemming is better than derivational. The inflectional stemmer produced an improvement of 7 % as compared to no stemming. In the case of French, an earlier attempt to generate stopwords list and stemming procedure has been made by Savoy (1999). Savoy proposed a weak stemmer which eliminates only the plural inflections of the word and does not consider variations on account of the person, tense, verbs, etc. This stemmer is similar to simple s-removal stemmer for English but the author suggested that to handle the French morphology, a more aggressive stemmer is required which can remove other variations in the words. Savoy (2006) also studied the effect of stemming on IR for non-English languages and verified that light stemmers are less effective than aggressive stemmers for Romanic languages. The author developed light stemmers for French, Portuguese, German and Hungarian languages and found an increase of 35, 22, 9 and 42.8 % respectively in retrieval performance. The stemmer performed comparable to Porter like stemmer for these languages. Majumder et al. (2007a) compared the Porter version of French stemmer with a statistical stemmer based on string similarity and reported that the differences in both the techniques are statistically insignificant.

For stemming in Portuguese, Orengo and Huyck (2001) developed a rule based suffix stripping stemmer that eliminates the suffixes using linguistic rules in eight steps such as plural, feminine, adverb, noun suffix, verb suffix, vowel and accent removal. The proposed stemmer is evaluated using three mechanisms including Paice (1994) mechanism and reported that it outperformed the Portuguese Porter stemmer (Porter 2001). Another stemmer STEMBR was proposed for Brazilian Portuguese by Alvares et al. (2005). It removes the suffixes from words using last common alphabet. STEMBR produced less under stemming errors and more over stemming errors as compared to Portuguese Porter stemmer. Soares et al. (2009) proposed an improvement in Portuguese Porter algorithm. The algorithm used the features and the problems of the Portuguese language. The improved version of Portuguese Porter algorithm is faster and has improved the quality of stemming. In Italian, the effectiveness of stemming and compound splitting has been studied by Monz and Rijke (2002). The authors reported consistent improvement of nearly 25 % in information retrieval over no stemming.

– *Stemming in Germanic languages* Besides English, most of the stemming work in the class of Germanic languages is reported for Dutch, Swedish, and German. For stemming

in Dutch, Kraaij and Pohlman (1994) designed a well known Dutch variation of Porter algorithm. The rules covered the inflectional morphology of language, noun, and adjective forming derivational endings and the most commonly occurring suffixes of Dutch. The algorithm is evaluated using Paice mechanism (1994) along with the information acquired from CELEX database Baayen et al. (1993) and is found to be consistent. The work has been further carried forward by authors in Kraaij and Pohlman (1996), in which inflectional, derivational and Porter stemmer for Dutch are tested using Hull mechanism (1996) for enhancement of recall. The authors reported that among the three stemmers, inflectional stemmer is the most successful one and the removal of derivational suffixes are sometimes useful but in most of the cases it decreases precision. Gaustad et al. (2002) proposed a hybrid Dutch stemmer that combines the dictionary lookup with Dutch Porter stemmer. In order to speed up the lookup process, FSA encoding of dictionary CELEX (Baayen et al. 1993) is used. The proposed stemmer has been used for text classification tasks and verified that stemming improves classification accuracy. For stemming in Swedish, Porter style algorithm has been implemented by Carlberger et al. (2001). The stemmer makes use of nearly 150 stemming rules that are applied in four steps. The author reported an improvement of 15 % in precision and 18 % in recall on Swedish data collection. In the case of German language, Braschler and RippLinger (2004) studied the effectiveness of stemming in information retrieval using a number of stemmers like Linguistica (Goldsmith 2001); German Porter stemmer (NIST); MPRO (Maas 1996), etc. The authors concluded that simple rule based stemmers like NIST can produce consistent results in German, but the major concern in German is to handle compound words. The decompounding of words can further improve the performance of stemming algorithm.

– *Stemming in Balto-Slavic languages* Balto-Slavic languages have fusional morphology. Slavic languages are quite conservative and homogeneous. In this category, most of the stemmers are developed for Slovene, Czech, Bulgarian and Russian languages. Popovic and Willet (1992) were the first who applied stemming to Slovene text collection and claimed that stemming produces insignificant improvement in IR for languages like English but for a language with complex morphology like Slovene the improvement is significant. For Bulgarian, an inflectional stemmer named *BulStem* has been proposed by Nakov (2003) that makes use of a morphological lexicon (CLPOI-BAS) containing different word forms and POS. The author reported efficiency between 60 and 80 % and proved that lemmatization and stemming are equally good for Bulgarian. Savoy and Berger (2006) developed stopwords list and light stemming approaches for Bulgarian, Hungarian and Portuguese languages. The stemmers remove the inflectional suffixes by considering the rich morphology of the languages. The stopwords and stemmers are used to evaluate the IR performance of various IR models using five vector scheme strategies. For stemming in Czech, Dolamic and Savoy (2009a) developed two stemmers, one light stemmer that removes only the inflectional suffixes and other an aggressive stemmer that also removes commonly occurring derivational suffixes. The authors claimed that stemming in Czech results in an improvement of nearly 45 % in IR and the results produced by aggressive stemmer are more promising than the light stemmer. The authors also developed light and aggressive stemmers for the Russian language in Dolamic and Savoy (2009b) and compared with the Russian version of Porter stemmer (available at http://snowball.tartarus.org) and n-gram (Mcnamee and Mayfield 2004) stemmer. In their evaluation, the light stemmer is reported to be the best, but the differences between the light, aggressive, Porter and n-gram are not statistically significant.
– *Stemming in Semitic languages* Semitic languages have fixed pattern of root words i.e. consonantal or trilateral, and the variants are formed by adding suffixes, prefixes, infixes,

vowels, doubling consonants. In the category of Semitic languages, most of the work has been reported for Arabic and Hebrew languages. Language-specific models of stemmers in Arabic are divided into two categories: stem-based (light) and root based (aggressive). Stem-based approaches remove only prefixes and suffixes attached to the words whereas root-based approaches consider infixes also along with prefixes and suffixes (Elrajubi 2013). The stem-based stemmers in Arabic are preferred in linguistic areas, and root-based stemmers are preferred in applications of information retrieval. A number of stemmers have been developed based on root-based approaches among which Khoja stemmer (Khoja and Garside 1999) is a popular Arabic stemmer. It is a rule-based stemmer that first removes the affixes and then matches the resultant word with a set of patterns of equal length to find the root. Another root-based stemmer has been described in Alshalabi (2005) that finds the tri-literal roots of the word by affix stripping and pattern matching. Taghva et al. (2005) identified three major problems in Khoja stemmer and proposed an improvement in it that does not make use of a dictionary and still achieved accuracy at par with Khoja stemmer. Kchaou and Kanoun (2008) also proposed an aggressive Arabic stemmer that makes use of two dictionaries (one for roots and other for stems). The proposed technique removes the problems of handicapped roots and stems in Khoja stemmer. Al-Kabi (2013) also identified some missing patterns in Khoja stemmer and introduced certain new patterns thereby improving the accuracy by 5 %.

The inception of Arabic IR track in TREC 2001 led to the development of a number of light stemmers. Larkey et al. (2002, 2007) developed a series of light stemmers (Light1, Light2, Light3, Light8, Light10) with minor variations. The stemmers are tested on TREC data along with some root-based stemmers and the Light10 stemmer outperformed among all the approaches. Chen and Gey (2002) developed a Berkley stemmer that is similar to Light10 stemmer but added more prefixes, suffixes and different constraints on the length of words. The authors claimed that Berkley stemmer outperforms Alstem (Darwish and Oard 2002) stemmer. Aljlayl and Frieder (2002) also proposed a light stemmer that is based on affix removal and normalization of words. The proposed approach performed better than no stemming and n-gram approach. Harmanani et al. (2006) devised a stem-based approach that makes use of rule engine which can be used for any language just by changing the grammar and rules according to the language. The system is fully implemented on Arabic dataset and partly on English, Persian and Hebrew languages. A dictionary free Educated Text Stemmer (ETS) has been proposed in Al-shammari and Lin (2008) that considers stopwords removal before performing stemming. El-Beltagy and Rafea (2011) extended light stemming approach by introducing the semi-automatic construction of stem lists from the corpus, ability to handle broken plurals and unsure affix elimination. Another improvement in light stemming approach has been suggested in Elrajubi (2013) in which recoding or normalization of roots is performed after prefix and suffix removal. The proposed stemmer is compared with Light10 stemmer and performed better.

– *Stemming in Indo-Iranian languages* Indo-Iranian languages include three subdivisions: Indo-Aryan, Iranian and Nuristani languages. A variety of stemmers have been developed for Urdu, Persian, and many Indian languages. Among Indian languages, we found the first lightweight Hindi stemmer UMass developed by Larkey et al. (2002) which makes use of 27 suffixes that are removed using longest match principle. The authors claimed that the use of stemmer did not improve cross-lingual retrieval by testing a small number of queries on 2927 documents collection. Another light weight stemming approach in Hindi has been proposed by Ramanathan and Rao (2003) that removes inflectional

suffixes according to longest match principle. The stemmer is tested by calculating over stemming and under stemming errors and achieved an accuracy of 82 %. Shrivastava and Bhattacharyya (2008) and Shrivastava et al. (2005) developed a rule based suffix stripping technique for Hindi that makes use of a dictionary. The stemmer is used to design an HMM-based POS tagger and achieved a tagging accuracy of 93 %. Light and aggressive stemming approaches are developed for Hindi, Bengali, and Marathi languages by Dolamic and Savoy (2010). Both the stemmers are evaluated using FIRE 2008 collections and aggressive stemmers performed better in all the languages. Ganguly et al. (2012) participated in Morpheme Extraction Task in FIRE 2012 and studied suffix stripping approaches for Hindi and Bengali. The Hindi stemmer removes plural inflections only and the Bengali stemmer also considers case markers and classifiers. For stemming in Bengali, a highly inflectional language, a variety of language dependent stemming approaches are proposed in the literature. Dasgupta and Khan (2004) developed a morphological analyzer that considers both the inflectional and derivational morphology of the language. Bhattacharya et al. (2005) developed a rule-based method for synthesis of inflectional stems for nouns, verbs, and pronouns. Another light weight stemmer has been developed by Islam et al. (2007) which make use of suffix list of 72 verbs, 22 nouns and 8 adjectives that are removed according to longest match principle. The stemmer is applied and evaluated for spell checking and achieved consistent performance. A cluster based stemming technique has also been described in Das and Bandyopadhyay (2010) that makes use of suffix stripping and score based feature (minimum edit distance) for clustering. Recently, Mahmud et al. (2014) developed a Bangla rule based stemmer that removes noun and verb inflections in a number of steps without using any dictionary.

Gujarati is another Indian language in which language-specific stemming approaches are reported in the literature. Patel et al. (2010) proposed a hybrid semi-automatic light weight stemmer. The stemmer is based on Goldsmith (2001) technique, but it makes use of manually generated suffix list. The stemmer used EMILE corpus (available at http://www.lancs.ac.uk/fass/projects/corpus/emille) for training and testing and achieved an accuracy of nearly 67 %. Suba et al. (2011) improved the light weight stemmer described in Patel et al. (2010) by using POS stemming and substitution rules and achieved an accuracy of nearly 90 %. For stemming in Punjabi, two rule-based stemmers are described in Kumar and Rana (2010) and Gupta and Lehal (2011) which perform suffix stripping using manually created language specific rules. A series of stemmers are developed for other Indian languages like Oriya (Chaupattnaik et al. 2012), Tamil (Ramachandran and Krishnamurthi 2012; Lushanthan et al. 2014), Kannada (Hegde et al. 2013; Deepamala and Kumar 2015), etc.

For stemming in Urdu, Akram et al. (2009) proposed an affix removal stemmer named Assas-Band. It removes prefixes and postfixes using two exception lists and then it adds one or more letters to find the root word. Another rule based stemmer for Persian has been developed by Sharifloo and Shamsfard (2008) that uses bottom–up approach and achieved an accuracy of nearly 90 %.

– *Stemming in Malayo-Polynesian languages* In this language family, the morphological variants are formed by reduplication (repeating some or whole word) and affixation (adding prefixes and suffixes). In Malayo-Polynesian languages, stemmers are developed for Malaysian and Indonesian languages. We found the first rule based Malay stemmer developed by Othman (1993) which makes use of nearly 121 rules to strip the suffixes, prefixes, and infixes. Ahmad et al. (1996) improved the Othman algorithm by using a dictionary or by correctly expanding the rules. Another affix stripping Malay stemmer is described in Tai et al. (2000). The stemmer is tested on some Malay web documents

and achieved considerable improvements in information retrieval. Sembok (2005) used stemming in Arabic and Malay documents to evaluate the retrieval effectiveness and verified that retrieval accuracy improves with stemming. For stemming in Indonesian, a Porter style stemmer is developed in Tala (2003) but the experiments did not report any considerable improvement in information retrieval.

- *Stemming in Uralic languages* Uralic languages are agglutinative languages which are highly inflective and rich in morphology. In this class, stemming approaches are developed mainly in Hungarian and Finnish languages. Korenius et al. (2004) developed stemming and lemmatization approach for Finnish language and evaluated both techniques using four different hierarchical clustering approaches. The author verified that lemmatization approaches are better than stemming in clustering Finnish documents. For stemming in Hungarian, Savoy (2008) has a major contribution in developing a light weight and aggressive stemming approaches that remove suffixes according to the rich morphology of the language. Based on the experiments performed on the CLEF document collection, authors claimed that aggressive stemming results in higher improvement as compared to light stemming approaches and the differences were statistically significant.
- *Stemming in Turkic languages* Turkic languages are another language family belonging to agglutinative languages. The languages belonging to this family are derived from complex morphology and are highly inflectional. In our study, we found FindStem (Sever and Bitirim 2003) was the first Turkish stemmer described in the literature. FindStem used a dictionary to find root word and achieved an improvement of 25 % as compared to no stemming. Due to complex morphology of the Turkish language, morphological analyzer described in Eryiğit and Adalı (2004) is used as stemmer in a number of information retrieval tasks. Cilden (2006) developed a Porter algorithm based stemmer for Turkish using Snowball language that removes suffixes using suffix list and a set of rules.
- *Stemming in Hellenic languages* Hellenic languages are fusional languages that include Greek alone. In our study, we found that Kalamboukis and Nikolaidis (1995) proposed the first suffix stripping stemmer for Greek which works in two phases to remove the inflectional and derivational suffixes. The stemmer is first evaluated on the computer and medical science documents and achieved satisfactory recall and precision. The evaluation is further extended by the same authors in (Kalamboukis and Nikolaidis 1999) using SMART system (Salton and McGill 1971) and various statistical tests. The Porter version of Greek stemmer is described in Ntais (2006) and is tested on two different document collections. The stemmer achieved an accuracy of nearly 90 % on both the data sets. In order to handle the complex morphology of Greek, Adam et al. (2010) proposed a stemmer that first performs POS tagging and then strips the suffixes according to the POS tagged to the word. The author claimed a high accuracy of 96.7 % in stemming the words.

Table 4 sums up the language-specific stemming algorithms covered within the scope of our survey. The table lists nearly 76 publications that are grouped into nine different language families.

4 Statistical stemmers

Statistical stemmers overcome the problems related to linguistic models as they are based on unsupervised learning of the language. A number of studies related to the unsupervised learning of morphology of the language are published in the literature. A comprehensive

Table 4 Language specific stemming approaches

Italic languages	
Spanish	Honrado et al. (2000), Figuerola et al. (2001)
Italian	Monz and Rijke (2002)
French	Savoy (1999, 2006), Majumder et al. (2007a)
Portuguese	Orengo and Huyck (2001), Alvares et al. (2005), Savoy (2006), Soares et al. (2009)
Balto-Slavic languages	
Slovene	Popovic and Willet (1992)
Bulgarian	Nakov (2003), Savoy and Berger (2006)
Czech	Dolamic and Savoy (2009a)
Russian	Dolamic and Savoy (2009b)
Germanic languages	
English	Lovins (1968), Dawson (1974), Porter (1980), Paice (1990), Krovetz (1993), Hull (1996)
Dutch	Kraaij and Pohlman (1994, 1996), Monz and Rijke (2002), Gaustad et al. (2002)
Swedish	Carlberger et al. (2001)
German	Monz and Rijke (2002), Savoy (2006)
Semitic languages	
Arabic	Khoja and Garside (1999), Larkey et al. (2002), Chen and Gey (2002), Darwish and Oard (2002), Aljlayl and Frieder (2002), Alshalabi (2005), Sembok (2005), Taghva et al. (2005), Harmanani et al. (2006), Larkey et al. (2007), Kchaou and Kanoun (2008), Al-shammari and Lin (2008), El-Beltagy and Rafea (2011), Elrajubi (2013), Al-Kabi (2013), Al-Zyoud and Al-Rabayah (2015)
Hebrew	Harmanani et al. (2006)
Indo-Iranian languages	
Persian	Harmanani et al. (2006), Sharifloo and Shamsfard (2008)
Urdu	Akram et al. (2009)
Hindi	Larkey et al. (2003), Ramanathan and Rao (2003), Shrivastava and Bhattacharyya (2008), Shrivastava et al. (2005), Dolamic and Savoy (2010), Ganguly et al. (2012), Gupta (2014)
Bengali	Dasgupta and Khan (2004), Bhattacharya et al. (2005), Islam et al. (2007), Dolamic and Savoy (2010), Das and Bandyopadhyay (2010), Mahmud et al. (2014)
Marathi	Dolamic and Savoy (2010)

Table 4 continued

Indo-Iranian languages	
Gujarati	Patel et al. (2010), Suba et al. (2011)
Punjabi	Kumar and Rana (2010), Gupta and Lehal (2011)
Oriya	Chaupattnaik et al. (2012)
Tamil	Ramachandran and Krishnamurthi (2012), Lushanthan et al. (2014)
Kannada	Hegde et al. (2013), Deepamala and Kumar (2015)
Malayo-Polenisyan	
Malay	Ahmad et al. (1996), Tai et al. (2000), Sembok (2005)
Indonesian	Tala (2003)
Uralic	
Finnish	Korenius et al. (2004)
Hungarian	Savoy (2008)
Turkic	
Turkish	Sever and Bitirim (2003), Eryiğit and Adalı (2004), Cilden (2006)
Hellenic	
Greek	Kalamboukis and Nikolaidis (1995, 1999), Ntais (2006), Adam et al. (2010)

and an outstanding survey is presented in Hammarström and Borin (2011), which gives a detailed comparison and explanation of techniques related to morphologies of the languages at various levels. In this section, several corpus-based statistical stemmers are described. Section 4.1 describes lexicon analysis based approaches which consider a set of words and try to determine the most likely roots and suffixes and Sect. 4.2 describes corpus analysis based approaches which associate lexicographically-identical words by considering the context or co-occurrence of words in the corpus. Section 4.3 describes various n-gram stemming approaches presented in the literature to handle morphology of alphabetic languages.

4.1 Lexicon analysis based approaches

– *Letter successor varieties* The idea of segmentation of words using some variations of letter successor variety counts is first used by Hafer and Weiss (1974) for stemming in information retrieval tasks. The words are segmented into roots and suffixes using different combinations of predecessor and/or successor variety counts with corresponding cutoff values; successor or predecessor counts being on a peak or plateau; the prefix/suffix being a complete word by itself at a specific location; and successor and predecessor entropy. The authors performed 15 different experiments on Carolina Population Center (CPC) and American Documentation Institute (ADI) English test collections and provided results for 13 of them. The best combination chosen by authors is a hybrid of cutoff (threshold) values for entropy scores or variety counts along with the knowledge on whether some part of the word appear as a standalone word by itself or not. Al-Shalabi et al. (2005) also applied the same combination on Arabic documents and reported that it works well for the Arabic Language as well.

Another variation of successor variety has been applied to the field of stemming in Stein and Potthast (2007). The successor variety stemming approach proposed in the paper relies on a tree structure of suffixes and is regulated by some pruning rules that examine the successor

variety of the inner nodes. The paper investigates the effect of successor variety stemming and rule-based Porter stemming on English and German collections (Reuter collection and German newsgroup postings). The authors verified that rule based stemming performed slightly better than successor variety stemming but it can be improved by slightly modifying the pruning rules to handle compound words in German.

– *Suffix frequency based stemmer* Oard et al. (2001) developed a stemming technique which discovered suffixes from the corpus statistically and removed it from the word endings. In this technique, from the first 500,000 words in the corpus, the frequency of every ending of length one, two, three and four that produces a root of length at least three characters are computed. The frequencies are then adjusted by subtracting the frequency of subsuming ending from the next longer length suffix. This is done to eliminate the partial suffixes as the suffix '-ng' is mostly part of the suffix '-ing'. In order to decide the number of endings to be produced for each length, a plot of count and rank is made and the point at which the second derivative of the plot is maximized is considered as the cut-off. The adjusted frequencies are then arranged in decreasing order for each length and are used to determine the most likely valid suffix. The proposed statistical stemming is used to implement a four-stage document backoff translation strategy, which achieved an improvement of 10 % in retrieval accuracy for multilingual documents. For French, 55 % improvement is reported whereas no significant improvement is reported in German and Italian probably due to inappropriate threshold selection criteria.

Paik and Parui (2011) proposed a fast and robust stemmer that also performs stemming on the basis of potential suffixes discovered from the corpus. It is based on the fact that suffixes occur quite frequently in a large corpus. In this technique, the potential suffixes are first identified according to their frequency in the corpus. The words are then clustered into classes according to common prefix, and the strength of each class is measured using potential suffix knowledge. The strength determines whether the common prefix string of the class is the stem or not. If not, then another root in the class is determined iteratively. The technique is implemented in four languages which are English, Hungarian, Bengali and Marathi using TREC, CLEF, and FIRE collections. In all languages, there is a significant improvement over no stemming. Yadav et al. (2012) used a similar kind of suffix frequency based stemmer in retrieval experiments on some Indian languages and achieved a significant improvement in Marathi, Bengali, Gujarati, and Oriya. But on English and Hindi test collections the results are not promising.

– *Minimum description length (MDL) based stemmer* An unsupervised stemming technique using Minimum Description Length (MDL) approach is described in Goldsmith (2001). The approach determines the optimal breakpoint (point of the optimal split of the word into stem and suffix) so that every word in the collection is divided into common suffixes and common roots. The technique proposed in the paper calculates the frequency of the root and suffix at every possible breakpoint for each word in the collection. The breakpoint is optimal if for every instance of the word the breakpoint is same and minimizes the necessary bits required to encode the text collection (which is analogous to reducing the entropy of the collection). The software framework for this approach is named as Linguistica (Goldsmith 2006). Goldsmith verified the effectiveness of automatic suffix identification in Information Retrieval by correctly stemming 83 % of words in both English and French, but the technique involves a lot of computations with initial implementation time for a moderate sized collection running into days.
– *Hidden Markov model based stemmer* The concept of Hidden Markov Models (HMM) is applied to the field of stemming by Melucci and Orio (2003). HMMs are automata in

which the probability functions decide the change from one state to another. These HMMs are used to model the words, and the letters of the words are considered as states. In order to model the stemming process using HMMs, the word is considered to be formed by concatenation of two strings—prefix and suffix. The states of the HMM are partitioned into two sets; one corresponds to the prefix that gives the first part of the word and the second corresponds to the suffix set that gives the ending part if any. For any given word, the HMM with maximum probability is chosen, and the change from prefix-set state to a suffix-set state in the chosen HMM determines the *split point*. The set of letters generated by the states before the *split point* corresponds to be the root of the word. The authors used three different topologies of HMMs in the experiments and compared the algorithm with classical Porter's algorithm in five different languages (English, Dutch, French, Italian, and Spanish). The retrieval accuracy of HMMs is reported to be equivalent to Porter's algorithms; in some cases HMM stemmer tends to over-stem the words.

– *Probabilistic model for stemmer generation* Bacchin et al. (2005) suggested a method that finds the best split by maximizing the probability of the prefix and the suffix pair. The proposed technique is an extension of author's previous work *Graph Based Stemming* described in Bacchin et al. (2002), as it models the mutual reinforcement between the stem and the derivation using the probabilistic framework. The method first finds a set of substrings by breaking each word in the collection at every possible point. Then the set of substrings is transformed to a directed graph where substring represents the nodes of the graph, and there is an edge from node u to v for every word w such that w = uv. The HITS link analysis algorithm (Kleinberg 1999) is then used to determine the prefix and suffix score based on the assumption that good prefixes point to good suffixes, and good suffixes are pointed to by good prefixes. The suffix and prefix score is then used to determine the best split by maximizing the likelihood of the prefix and suffix pair. The authors performed a number of experiments to compare the retrieval efficiency of stemmer with Porter stemmer in some languages (English, Italian Dutch, etc.) and reported that both the stemmers perform equally well in all the languages.
– *String similarity based word clustering* Conflation is viewed as a word clustering problem in Majumder et al. (2007b) to develop a language-independent stemmer named Yet Another Suffix Stripper (YASS). The method makes use of certain string distances that do not have any knowledge of morphological variants of language and gives high similarity between words that have a long common prefix. For any two strings X and Y of length n + 1 (null characters are appended to the shorter string to make length equal), the authors proposed four string similarity measures $\{D_1, D_2, D_3, D_4\}$ and reported that D_3 (as given in equation 1) is the most effective and is quite insensitive to the changes in threshold values.

$$D_3(X, Y) = \frac{n - m + 1}{m} \sum_{i=m}^{n} \frac{1}{2^{i-m}} \quad \text{if m} > 0; \quad \infty \text{ otherwise} \tag{1}$$

where m is the location of the first mismatch between the strings X and Y. After calculating the string distance parameters, the clusters are obtained using graph based complete linkage technique due to its ability to produce compact classes and find natural clusters in the data. The number of clusters of the lexicon is determined by a threshold value which must be carefully chosen so that equivalence classes produced are appropriate, otherwise too aggressive or too lenient stemmer is constructed. The proposed stemmer can handle a family of suffixing languages. The authors showed that the proposed approach performed equivalently to rule based Porter and Lovins stemmer for English. For Bengali

and French, the approach achieved a significant improvement over no stemming. YASS is also implemented on Hungarian and Czech languages in Majumder et al. (2008). For both languages, the stemmer is reported to be equivalent to the corresponding rule-based stemmers.

Fernández et al. (2011) also proposed an unsupervised algorithm that automatically creates morphological classes using Extended Edit distance (EDx). The distance measure considered the characteristics related to the etymology (word derivation history) to improve the morphological classes. The penalization introduced in this method allowed orthographic susceptibility. Baroni et al. (2002) also developed a method of unsupervised stemming using orthographic similarity through edit distance and semantic features like mutual information.

– *Graph-based stemming* Paik et al. (2011a) used weighted graphs to build a stemmer named GRAph based Stemmer (GRAS). The stemmer discovers frequently occurring suffix pairs, rather than single suffix from the corpus. A weighted undirected graph is then constructed using the knowledge of suffix pairs whose nodes are the words of the corpus, and there is an edge from node x to y if both words induce a suffix pair whose frequency is above some pre-decided threshold. The frequency of the suffix pair is assigned as the weight of the corresponding edge. The graph is then decomposed by computing cohesion between the node with a maximum degree (pivotal node, p) and nodes adjacent to it in decreasing order of edge weights according to equation (2). It is based on the assumption that if there are many neighbors of the word, then it is probably the root. In each step, a new morphological family is identified and deleted from the graph. The process continues until all the vertices are processed. GRAS is found to be very effective in Information Retrieval tasks in a number of European and Asian languages and outperforms other methods belonging to this category. It uses an efficient graph-based algorithm coupled with a common prefix and suffix pair (rather than single suffix) information, which more accurately forms the equivalence classes thereby improving the retrieval performance.

$$Cohesion\ (p,q) = \frac{1 + |Adjacent\ (p) \cap Adjacent\ (q)|}{|Adjacent\ (q)|} \tag{2}$$

4.1.1 Distinguishing features of lexicon analysis based statistical stemmers

Various lexicon analysis based stemmers discussed in this section are dependent only on the lexical knowledge about the words. Unlike corpus analysis based stemmers, these stemmers can work well on all the corpus of corresponding language whereas corpus analysis based stemmers can only be employed on comparatively large document collections of that specific language so as to obtain dependable co-occurrence information (Paik et al. 2013). In this subsection, we discuss some of the distinguishing features of stemmers belonging to lexicon analysis based category that leads to differences in their experimental performances.

Oard et al. (2001) stemmer and fast corpus-based stemmer (FCS) (Paik and Parui 2011) work on the same principle of identifying potential suffixes on the basis of their frequency in the corpus. The major difference in both the stemmers is that Oard stemmer is based only on discovering potential suffixes from the corpus, but FCS stemmer considers common prefix information along with potential suffixes. The performance of FCS stemmer is, therefore, better than Oard in retrieval experiments. Both the stemmers are fast, computationally less intensive and robust in terms of parameters selection. But, the major flaw in both the stemmers is that these stemmers ignore some potential suffixes which are infrequent. Due to which

these stemmers have weak strength as there are comparatively less number of words in each conflation class.

The minimum-description-length (MDL) based framework Linguistica proposed by Goldsmith (2001) effectively breaks the words into stems and suffixes. The major advantage of MDL based morphological analyzer is that it can handle out-of-vocabulary words as it identifies when one signature's endings are a proper subset of another (Chan 2006). The computational overhead of the Linguistica is very high as compared to all other stemmers of this category. The stemmer proposed by Melucci and Orio (2003) also effectively finds optimal breakpoint between the prefix and the suffix through maximum likelihood estimation of probabilities using HMMs. The performance of HMM-based stemmer is not only dependent on the frequency of suffixes, but the most probable path is dependent on modeling of both suffixes and prefixes. Moreover, HMM-based stemmer can also handle compounding of words (i.e. "moonlight") and spelling variations (i.e. "lovable" and "loveable"). The stemmer described in Bacchin et al. (2005) also uses the probabilistic framework to identify potential prefixes and suffixes from the corpus through mutual reinforcement relation between them. But, it is not only based on a calculation of the frequency of sub-strings as in the case of successor variety based (Hafer and Weiss 1974), MDL based (Goldsmith 2001) or HMM-based (Melucci and Orio 2003) stemmers but also dependent on link analysis methods which are quite effective in web retrieval.

YASS (Majumder et al. 2007b) and stemmer described in Fernández et al. (2011) are both based on clustering of words on the basis of string distances. The former stemmer uses a new set of string distances suitable for stemming that avoids early mismatch and awards longest common prefix while the later uses a variation of well-known edit distance. YASS is developed to handle languages with concatenative morphology and is universal in the sense that it performs well in IR tasks, inflection removal experiments, and language modeling tasks (Brychcín and Konopík 2015). The major problem in YASS is that complete linkage clustering algorithm fails to handle large equivalence classes formed in highly inflectional languages where large numbers of suffixes are present. A large equivalence class is partitioned into a number of small classes, and hence, its performance degrades in languages with complex morphology like Hungarian and Marathi. Moreover, the complete linkage algorithm also depends on the order in which the objects are considered which sometimes results in the formation of inaccurate clusters.

Graph-based stemmer (GRAS) (Paik et al. 2011a) maintains a good balance between complexity and efficiency. It shows high retrieval efficiency as compared to other stemmers of this category. But, it is highly focused on recall rate and thus tends to over-stem the words.

Among all the lexicon analysis based stemmers discussed in this section, Oard and FCS are fastest as they take few minutes to construct from a large corpus, followed by GRAS, which takes slightly more time than the two. YASS, HMM-based and Probability based stemmers, on the other hand, are almost same in terms of computational intensity and takes hours to complete the job. Goldsmith's Linguistica is slowest among all and takes days to construct from a moderate sized corpus. In terms of strength i.e. mean number of words per conflation class, YASS and GRAS are most aggressive with a maximum number of words per conflation class. Linguistica, HMM-based and FCS stemmers have moderate strength whereas Oard is the lightest among all with least number of words in each conflation class.

4.2 Corpus analysis based approaches

– *Corpus analysis using word co-occurrences* In our study, we found that Xu and Croft (1998) first used corpus analysis approach to refine or to develop a stemmer. The method

is based on the presumption that variants occur in the same text or, more precisely in the 100 words text window. In order to calculate the association between the word pairs, the authors used a variation of Expected Mutual Information Measure (EMIM) as given in Eq. (3).

$$em(w_a, w_b) = \max\left(\frac{n(a,b) - kn_a n_b}{n_a + n_b}, 0\right) \tag{3}$$

where n(a, b) is number of co-occurrence of words a and b, n_a and n_b are number of times words a and b occur in the documents respectively, k is a corpus dependent parameter. After calculating the EM scores, the equivalence classes already created by some aggressive stemmer like Porter are upgraded using connected component and/or optimal partitioning graph clustering algorithms. The authors refined the equivalence classes created by a few rule based stemmers on English and Spanish data collections and reported improvement in recall. The corpus-analysis based method helped in correcting many wrong conflations (universe and university) made by language-specific stemmers by identifying conflations in the corpus.

A similar kind of approach is proposed by Paik and Parui (2011) which makes use of a simpler co-occurrence measure and a novel nearest neighbor based clustering approach. The clustering approach does not require any parameter tuning as in the case of other clustering algorithms like single or complete linkage clustering. The method first groups the words that share a common prefix and then calculate co-occurrence between a pair of words in each class using Eq. (4)

$$CO(u, v) = \sum_{d \in C} \min(\mathrm{f_{u,d}}, \mathrm{f_{v,d}}) \tag{4}$$

where $\mathrm{f_{u,d}}$ means the frequency of document u in d. The co-occurrence information is then mapped into weighted undirected graph whose nodes are the words of the collection, and there is an edge between two words if the co-occurrence between them is greater than zero and the weights of the edges being the co-occurrence strengths. These strengths are then recalculated using nearest neighbor information and are used to identify the strong edges. Finally, the equivalence classes are obtained using the connected component technique. The proposed technique is implemented on four European and two Indian languages. In the case of Czech, Marathi, Hungarian and Bulgarian more than 50 % improvement is reported and in Bengali and English nearly 20 % improvement is reported.

- *Distribution similarity based stemming* Bhamidipati and Pal (2007) proposed a technique for stemmer refinement that considers classification knowledge of the corpus. It assumes that the documents in the corpus belong to different categories, and the words are assumed to belong to multinomial distribution over the various categories of the corpus. These words are stemmed on the basis of their distribution similarity over various document categories where the distribution of each word is calculated from the frequency of occurrence in all the categories. The authors showed that refined stemmers increase the precision as compared to existing stemmers on Wall Street Journal collections. Moreover, the superiority of the refined stemmers is indirectly evaluated for classification purposes using classifiers like SVMs, Naïve Bayesian.
- *Context sensitive stemming* Peng et al. (2007) pointed out that blind stemming during query expansion does not consider the context in which the term is used and performs blind comparisons of all the occurrences in the documents. To overcome these problems, a context sensitive stemming approach is proposed that performs stemming both at query and document side. At the query side, the important variant words for query terms are

decided by using bigrams as context characteristics to the left and the right of the words. At the document side, conservative document matching is performed in which only those matches of the variant words are correct which occur in the similar context as that of the original word. The technique improves query traffic and is reported to be quite efficient on web data, but as it considers variants given by Porter stemmer, so performance is degraded due to under stemming errors.

– *Query based stemming* Paik et al. (2013) proposed a query based stemming method which provides those variants of the words that are thematically coherent with the words of the query. The authors developed query-independent and, more particularly, query-dependent stemming techniques. The query-independent technique first identifies candidate word pairs using the concurrence statistics and structural similarity knowledge and then it forms classes of morphologically related words. The method is based on the direct relation between the query terms and its candidate word variants. For each word of the query, one class of variant words is produced. Each word in the class is treated as being the same (using *syn* operator); therefore, each document term frequency for the class is the sum of term frequencies of individual words and the collection term frequencies is the sum of collection term frequencies of the individual words. The following example shows the output query produced by the query-independent stemming algorithm for the input query "education standards".

```
Output query: <query>
              #syn (educate educates educated educator education educations educating educational educationist)
              #syn (standard standards standardize standardizes standardized)
             </query>
```

The query-dependent stemming algorithm considers the morphological variant words for each query term from a base stemmer and reduces the effect of those variants that are not coherent with the original words of the query using co-occurrence metrics namely *Weighted Mean Association (WMA) or Association-Replacement Trade-Off (ART).* These co-occurrence metrics assigns weight to each variant term of the query word. The query terms are thus represented as an ordered pair of weight and term using a *wsyn* operator which allows the terms to be weighted during query and documents matching. Therefore, the document term frequency for each class is the weighted sum of term frequencies of individual words and similarly, the collection frequency is the weighted sum of collection term frequencies of individual words. The following example shows the output query produced by the query-dependent stemming algorithm for the input query "it education standards".

```
Output query: <query>
               #wsyn (0.19 educate 0.15 educates 0.10 educated 0.44 educator 1.0 education 0.45 educations
                     0.32 educating 0.70 educational 0.52 educationist)
               #wsyn (0.85 standard 1.0 standards 0.22 standardize 0.10 standardizes 0.11 standardized)
               </query>
```

The authors verified the effectiveness and robustness of query-dependent stemming on TREC, CLEF, and FIRE collections and compared the performance of query based stemming with stemmers by Porter (1980), Xu and Croft (1998), and Paik and Parui (2011); Paik et al. (2011a). The performance of query based stemmer using *ART* is reported to be remarkably better than all other stemmers in all the languages.

– *High precision stemmer* Brychcín and Konopík (2015) proposed an unsupervised stemming algorithm with the goal of developing a multi-purpose tool that can perform tasks other than traditional information retrieval such as removing inflections or improving

language modeling. The proposed stemmer increases precision by providing accurate stems at the expense of a very little decrease in recall. The stemmer is built up from the corpus in two stages. During the first stage, the words are clustered into different classes that share a common prefix and occur in similar context. The clustering algorithm is based on Maximum Mutual Information (MMI) which exploits the lexical and semantic information of the words. The second stage considers the clusters generated in the first stage as training data for maximum entropy classifier which provides stemming decisions that when and how to stem the words. The stemmer is compared with three strong statistical stemmers (GRAS, YASS, and Linguistica) and one rule based stemmer in six different languages belonging to four language families. The stemmer excelled in stemming the unseen words which are not present in the training collection.

4.2.1 Distinguishing features of corpus analysis based statistical stemmers

The corpus analysis based stemmers described in this section exhibit certain features of the ambient corpus. They use context or co-occurrence information of the corpus to analyze the morphology of the language. This analysis is found to be better than blind linguistic-stemmers especially in precision oriented systems (Xu and Croft 1998; Peng et al. 2007).

The stemmer proposed by Xu and Croft (1998) and co-occurrence statistic based stemmer (SNS) described in Paik and Parui (2011) both cluster words on the basis of co-occurrence information. The former uses a variant of expected mutual information while the later proposed a simpler co-occurrence strength measure based on term frequency of co-occurrence of two words. The co-occurrence measure used in XU is dependent on some corpus-based parameter and therefore requires parameter tuning whereas SNS is insensitive to parameters. Moreover, the connected component clustering mechanism used in XU forms chains (long equivalence classes). It increases recall, but there is a decrease in precision which eventually degrades mean average precision. The other optimal partition clustering algorithm used in XU cannot handle large size equivalence classes and is unsuitable for languages with complex morphology. SNS, on the other hand, accurately cluster words using graph based nearest neighbor clustering mechanism combined with re-computed co-occurrence strength.

The stemmer described in Bhamidipati and Pal (2007) is used to refine equivalence classes of an existing stemmer using advanced co-occurrence information i.e. distributional similarity among the words. It uses classification knowledge of the text collections. The classes of the base stemmer are refined by considering various spelling variations, decompounding or words with multiple contexts. The context sensitive stemmer described in Peng et al. (2007) and query-based stemmer (QBS) (Paik et al. 2013) are useful in retrieval and searching applications as they do not blindly transform each term of the query. They lessen the effect of those variants which are not related to the original query content thereby reducing over-stemming errors. Other corpus-based stemmers, group many unrelated words due to the frequent occurrence of words in the corpus. Due to this many irrelevant documents are retrieved. But, the query based stemmers reduce the effect of these influences by using query specific knowledge.

High precision stemmer proposed by Brychcín and Konopík (2015) performs stemming using both the semantic and lexical information of words from the corpus. It is a multi-purpose stemming tool which performs well in information retrieval and other tasks such as language modeling, approximating lemmas, etc. Unlike other corpus analysis based stemmers, it does not need large corpus for training and gives satisfactory results with only 50,000 tokens. This feature of the stemmer is very useful for languages with low resources.

4.3 Character N-gram-based approaches

Mcnamee and Mayfield (2004) used character n-gram tokenization as an alternate technique for stemming. As compared to other statistical approaches, n-gram technique can not only handle inflectional and derivational suffixes but can also handle compound words and spelling exceptions. In order to apply character n-gram approach to stemming, n-grams i.e. sequence of adjacent characters of length n is chosen from the various words in the documents. The similarity between the n-grams is calculated considering the fact that the unique stems have less frequency whereas the morphological variants i.e. suffixes or prefixes have a comparatively higher frequency. The authors reported that 4 or 5 grams are well suited for most of the European languages. One of the major shortcomings of this technique is that n-grams result in large inverted index size. For p characters, the number of n-grams are $p - n + 1$, but only $(p + 1)/(m + 1)$ words; where m is the mean length of the word of a particular language. Due to increased index size, n-gram approach is approximately eight times slower and takes six times storage space as compared to simple indexing approach. The authors compared the retrieval performance of n-gram approach with no stemming and Porter's algorithm in eight European languages and reported that it performs better than unstemmed words but underperforms Porter algorithm in most of the cases. Mayfield and Mcnamee (2003) also suggested a pseudo stem approach (selecting single n-gram for a word) to overcome the performance issues of character n-gram tokenization.

Another variation of unsupervised n-gram approach named Swordfish is described in Jordan et al. (2006). It identifies the roots of the words by calculating n-gram probabilities in the corpus. The method first calculates the frequency of n-grams of all lengths in the corpus. The probability of all the n-grams is then calculated as the ratio of the frequency of the n-gram to the total of frequencies of n-grams of all sizes. In the next step, the best split of the word is decided by using either joint probabilities or log odds between prefix probabilities.

Ahmed and Nürnberger (2009) pointed out that the pure n-gram approach for stemming did not take into account the order of various n-grams in the word. Due to which the similarity between the strings is high even if they do not share identical concepts. Moreover, the authors pointed that for highly inflectional languages like Arabic, different length words also share identical concept. The authors, therefore, proposed a refinement of pure n-gram approach that limits the search to certain fixed locations of the word by considering the order of n-grams. The algorithm is proposed for bigrams and can be extended to trigrams or n-grams.

Table 5 summarizes various statistical methods employed for development of corpus-based stemmers. The table also highlights pros and cons of the stemmers along with their experimental results.

5 Evaluation: metrics, results, and analysis

The strength and effectiveness of stemming algorithms are measured through a variety of mechanisms and is always a debated affair. In this section, we discuss various evaluation metrics and experimental results described in the literature.

5.1 Evaluation metrics

The evaluation mechanism proposed in the literature can be classified as direct or indirect (Smirnov 2008). Direct methods evaluate the stemmers by calculating certain features based on errors, compression factors, conflation percentage, statistical significance tests, etc. Direct

Table 5 Statistical stemming techniques

S. no.	References	Type of technique	Technique used	Pros and Cons	Results
1	Hafer and Weiss (1974)	Lexicon analysis	Letter successor variety	**Pros** Stemming technique is based on structural linguistics due to which accurate boundaries of roots and stems are detected. **Cons** High number of corpus dependent tunable parameters, high complexity	74 % words in CPC and 61 % words in ADI collections are stemmed correctly
2	Xu and Croft (1998)	Corpus analysis	Graph clustering based on cooccurrence measure	**Pros** Corrects bad conflations made by rule-based stemmers by identifying corpus dependent conflations. **Cons** The connected component method creates chains, and optimal partitioning method is not suitable for languages with complex morphology	Improved precision up to 10 % by refining equivalence classes generated by Porter and n-gram stemmers on English and Spanish collections
3	Oard et al. (2001)	Lexicon analysis	Suffix frequency based	**Pros** Computationally very simple and fast. **Cons** Inappropriate threshold learning and selection criteria	10 % improvement in MAP in multilingual four stage backoff translation
4	Goldsmith (2001)	Lexicon analysis	MDL approach	**Pros** Requires no parameter tuning or manual processing. **Cons** Computationally intensive with large computation time	85.9 % precision and 90.4 % recall for English test collection. 86.9 % precision and 88.9 % recall for French test collections
5	Melucci and Orio (2003)	Lexicon analysis	Hidden Markov model (HMM)	**Pros** Unsupervised parameter calculation (using EM), requires small training data. **Cons** Complex implementation and tendency to over stem the words	5–18 % improvement in AvP over no stemming in five languages English, Spanish, French, Dutch and Italian. The performance of HMM and Porter is found to be equivalent
6	Mcnamee and Mayfield (2004)	Character N-gram analysis	Character n-grams frequency	**Pros** Can handle compound words, spelling variations and inflectional and derivational morphology. **Cons** Increase in size of the inverted index, large disk space, and retrieval time	n-gram outperformed over no stemming (10–45 % improvement in MAP) but underperformed Porter stemmer in 8 European languages

Table 5 continued

S. no.	References	Type of technique	Technique used	Pros and Cons	Results
7	Bacchin et al. (2005)	Lexicon analysis	Probabilistic model	**Pros** Fully automated require no parameter tuning or preprocessing. **Cons** Complex implementation and involves lot of computations	Nearly 10 % improvement in AvP over no stemming for Italian collection. Probabilistic stemmer performed equivalent to Porter for English, Italian, Dutch, French and German
8	Majumder et al. (2007b)	Lexicon analysis	Complete linkage clustering based on string similarity	**Pros** Simple, easy to implement, aggressive stemming with a large number of words in each conflation class. **Cons** String measure flatters for lengthy suffixes, manual thresholds tuning	4.5–15 % improvement in AvP over no stemming in English, French and Bengali languages. For English and French, YAAS performs equivalent to Porter
9	Peng et al. (2007)	Corpus analysis	Context sensitive stemming	**Pros** Does not perform blind stemming during query expansion and considers context features at query and document side. **Cons** Considers variants given by Porter stemmer, so stemming is not fully automatic and have under stemming errors	For 29 % of query traffic, DCG improved by 6.1 % on the queries and overall query traffic improved by 1.8 %
10	Bhamidipati and Pal (2007)	Corpus analysis	Distribution similarity based	**Pros** Improves existing stemmers by considering words contexts, substitute words(varying styles and spellings) **Cons** Not fully automatic stemming technique as it refines the equivalence classes created by the background stemmers	Significant improvement (up to 25 %) in recall and precision on WSJ by refining Porter and trunc3 stemmers
11	Paik et al. (2011a)	Lexicon analysis	Weighted undirected graph based	**Pros** Retrieval effectiveness, low computational cost. **Cons** Highly focused on recall rate and tend to overstem words in some cases, manual parameter tuning	More than 50 % improvement in MAP over no stemming in some languages. GRAS outperformed YASS, Linguistica, Oard, and rule based in seven languages

Table 5 continued

S. no.	References	Type of technique	Technique used	Pros and Cons	Results
12	Paik and Parui (2011)	Lexicon analysis	Equivalence classes based on suffix frequency and common prefixes	**Pros** Fast, low computational cost, robust in parameter selection. **Cons** Ignores infrequent potential suffixes in some cases, (suffix with frequency below cut-off)	15–60 % improvement in MAP over no stemming in English, Marathi, Bengali, and Hungarian. Proposed method performed equal to rule based and performed better than YASS (except in Bengali)
13	Paik et al. (2011b)	Corpus analysis	Nearest neighbor method based on cooccurrence metric	**Pros** Nearest neighbor along with recalculated cooccurrence metric creates accurate clusters, static parameters. **Cons** Re-weighting leads to increase in computations	More than 50 % improvement in Czech, Hungarian, Bulgarian and Marathi and nearly 20 % improvement in Bengali and English over no stemming
14	Paik et al. (2013)	Corpus analysis	Query dependent stemming using cooccurrence measure	**Pros** Provides those variants of the query words which are thematically coherent with original query words	Up to 30 % improvement in MAP over no stemming in English, Marathi, and Bengali and 50 % improvement in Czech. QBS outperformed co-occurrence based stemmers by Xu and Croft (1998), Paik and Parui (2011), GRAS and rule-based stemmer
15	Brychcín and Konopík (2015)	Corpus analysis	Modified MMI clustering and maximum entropy classifier	**Pros** Requires little training data as compared to other statistical stemmers, high precision at the cost of a small decrease in recall	Nearly 50 % improvement in MAP over no stemming in Czech and Hungarian and nearly 15 % improvements in English and Spanish. Stemmer outperformed GRAS, YASS and Linguistica in complex languages like Czech and Hungarian

evaluation methods are complex and tedious as they require test sets in advance. Indirect evaluation methods measure the performance of the stemming algorithms indirectly through information retrieval effectiveness. Indirect methods are highly dependent on the nature of the corpus and queries used during evaluation. The various direct and indirect methods of stemmer evaluation proposed in the literature are listed below:

5.1.1 Paice evaluation mechanism

Paice (1994) labels the performance of the stemmer irrespective of the task performed. The mechanism proposed by Paice considers the under-stemming and over-stemming errors returned by a stemmer.

- *Under stemming* It refers to the word pairs that have the same stem but are not clubbed together. For example, the words *adhere* and *adhesion* are not grouped together by the Porter algorithm, but they belong to the same stem. Paice suggested that under stemming errors can be measured in terms of under stemming Index (UI) given by $UI = 1 - CI$; where CI is Conflation Index and is defined as the ratio of the number of word pairs that are successfully grouped together to the total number of words.
- *Over stemming* It refers to those word pairs that have different stems but are grouped together. For instance, the words *probe* and *probable* are grouped together, but they should not be in the same class. Paice provided Over-Stemming Index (OI) as a metric to measure over-stemming errors which is given by $OI = 1 - DI$; where DI is called Distinctness Index and is defined as the ratio of the number of equivalent word pairs not grouped together to the total number of words. The ratio of under stemming Index to over stemming Index is termed as *Stemmer weight (SW)*. A low value of *SW* represents weak stemming algorithm whereas a high value corresponds to a strong stemmer. Paice compared three classical stemmers Lovins (1968), Porter (1980) and Paice/Husk (1990) using *SW* and reported that Paice/Husk stemmer is the strongest whereas Porter stemmer is the weakest. The same results are later on supported by Frakes and Fox (2003).

5.1.2 Hull evaluation mechanism

Hull (1996) pointed that if the difference between the performance of stemmers is small, then it is very hard to say that the difference is significant or not. In such cases, statistical hypothesis testing can provide useful information about the experimental results. Statistical testing is more important in stemming tasks because the queries used during testing are small samples of all the desirable queries. The statistical techniques can be used to check that the difference between the means is significant, or it is just by chance. He advocated that techniques like analysis of variance (ANOVA) can be applied if the sample is large, continuous and normally distributed.

5.1.3 Accuracy/strength measurement

It is a direct method of stemmer evaluation in which the output produced by the stemmer is compared with the desired outputs. Through this comparison, the accuracy or strength of the stemmer is measured as a percentage of correctly produced stems. Frakes and Fox (2003) also proposed some ways to determine the strength of the stemmer:

- *Index compression factor (ICF)/conflation rate* It indicates the reduction in the corpus size that is achieved through stemming. It is the degree to which the unique words in the documents are stemmed. A number of experimental studies (Lennon et al. 1981; Harman 1991; Paice 1994; Frakes and Fox 2003) have verified that large value of this parameter indicates stronger stemmer as they stem a large number of words.
- *Mean number of words per conflation class* It means a mean number of words that belongs to the same root word. For instance, if the conflation class of root disturb is {"disturbs", "disturbance", "disturbing", "disturbed"} then number of words in this conflation class is 4. Strong stemmers have more number of words in each conflation class as compared to weak stemmers.
- *Mean number of characters removed* The strength of the stemmers can also be measured by the number of characters removed by the stemmer as a strong stemmer removes more characters as compared to a weak stemmer. This measure does not consider the changes in the stem endings.
- *Difference between number of words and stems* Some words are themselves root words and are thus unchanged by stemmers. The strength of stemmer can also be determined as how frequently the stemmer leaves the word unaltered. A strong stemmer changes the words more frequently as compared to weak stemmers.
- *Mean and median modified hamming distance between words and their stems* Hamming distance between two words of the same length is the number of corresponding characters that are different. The Modified Hamming distance between the word and their stem is calculated by adding Hamming distance to the difference in the length of the word and its stem.

5.1.4 Evaluation of stemmer using information retrieval systems

It is an indirect method of stemmer evaluation wherein stemmer is used as a pre-processing tool of an IR system. This mechanism is used by the majority of stemming algorithms. The Information Retrieval process is composed of two sub-processes: *Indexing* and *Retrieval*. During the indexing phase, the documents in the corpus are represented by a set of keywords and usually follow the following steps: tokenization; stopwords removal; stemming; phrase recognition; and term weighting (Majumder et al. 2007b). During the retrieval phase, the keywords in the query are matched with the document's keywords and the relevant documents are retrieved in decreasing order of their ranks. Hence, the effectiveness of stemmer can be evaluated by measuring the increase in number of relevant documents retrieved. In an IR system the performance of stemmer is measured indirectly through following parameters:

- *Precision* It is the number of relevant documents is to the total number of documents retrieved by a retrieval system. The high value of precision indicates that relevant documents retrieved are more than the irrelevant documents. So, the high value of precision indicates a higher efficiency of the stemmer. Precision considers all the retrieved documents by the system, but it can also be calculated by considering documents up to a particular rank which is termed as precision at n i.e. P@n.
- *Recall* It is the number of relevant documents retrieved is to the total number of relevant documents. The high value of recall means more number of relevant documents are retrieved. Stemming is considered to be recall enhancing instrument as through stemming a large number of variant words match to a single word, due to which more documents match the terms of the query and hence more relevant documents are retrieved.

- *F-Score* F-Score considers both precision and recall to test the retrieval accuracy. It is the weighted mean of precision and recall. F-Score does not consider true or false rate and is extensively used in information retrieval areas such as query classification, document classification, machine learning, etc.
- *Average precision (AvP)* It is the mean value of precision and recall in the ranked set of documents. The precision, recall and F-Score are used for an unordered list of documents when no ranks are given. In a ranked list, the precision and recall values are calculated at every point and then average precision is calculated as the mean of P(r) from rank 1 to n.
- *Mean average precision (MAP)* It is defined as the ratio of the sum of average precision of all the queries to the total number of queries. It calculates average precision by using rankings from a number of queries.

Among a number of information retrieval systems, TERRIER (Ounis et al. 2006) is widely used in stemming experiments. TERRIER (available at http://terrier.org) is written in Java and is an open source IR system developed by the School of Computing Science, University of Glasgow. Porter stemmer (1980) is the default stemmer used in TERRIER. TERRIER is highly flexible, efficient and comprehensive platform for carrying out research using standard TREC, FIRE, and CLEF collections. Most of the weighting models such as BM25, DFR, TF-IDF, and LM are included in TERRIER. It internally uses UTF and thus provides support for corpora in many languages. Besides TERRIER, there are a number of alternative IR systems SMART (Salton and McGill 1971), Lemur/Indri (http://www.lemurproject.org), Lucene/Solr (http://lucene.apache.org), Xapian (http://xapian.org) to name a few dominant options. All of these provide basic IR tasks, and one can choose from them on the basis of programming language, IR model, and weighting scheme.

5.2 Evaluation results and analysis

In this subsection, evaluation results of certain linguistic and statistical stemming techniques surveyed in this paper are presented. On the basis of deep analysis of retrieval experiments in the literature, the performance of stemmers is reported on shared TREC, CLEF, and FIRE collections. IFB2 model is used in retrieval experiments, which is suggested to be one of the best-performing models. The model uses a combination of tf and idf along with two different stages of normalization. All results are obtained using TERRIER IR system. Table 6 lists the techniques used in the analysis of various results. We used mean average precision (MAP), recall-precision (R-P), Precision@10 (P@10) and number of relevant documents retrieved (Rel. Ret.) as evaluation metrics to compare the techniques. In the retrieval experiments for each language, the query-wise performance (on the basis of average precision values) of the best performing stemming method is compared statistically with other methods. The p values less than 0.05 in all the statistical tests denote superiority of one stemming method over the other.

5.2.1 Test collection

In order to widen the scope of analysis, the results are compared for multiple languages of different origin and complexity. Besides English, Czech and Hungarian languages are considered from European languages, Bengali and Marathi are considered from Asian languages. Hungarian and Marathi languages have complex morphology whereas English has the simplest morphology among all. Table 7 summarizes the document collection and queries used in the analysis. For English, two test collections are considered. First test collection includes

Table 6 Summary of techniques and abbreviations used in analysis

Abbr.	Meaning
LOVIN	Lovins stemmer (Lovins 1968)
PORT	Porter stemmer (Porter 1980)
XU	Co-occurrence based stemmer (Xu and Croft 1998)
OARD	Suffix discovery based stemmer (Oard et al. 2001)
HMM	Hidden Markov model based stemmer (Melucci and Orio 2003)
4-gram	Character n-gram stemming approach (Mcnamee and Mayfield 2004)
LING	Linguistica framework for MDL approach Based Stemmer (Goldsmith 2006)
YASS	Yet another suffix stripper (Majumder et al. 2007b)
RB-HU	Rule based stemmer for Hungarian (Savoy 2008)
RB-CZ	Czech rule based stemmer (Dolamic and Savoy 2009a)
RB-MA	Marathi rule based stemmer (Dolamic and Savoy 2010)
RB-BE	Bengali rule based stemmer (Dolamic and Savoy 2010)
GRAS	Graph-based stemmer (Paik et al. 2011a)
FCS	Fast corpus-based stemmer (suffix discovery based) (Paik and Parui 2011)
SNS	Co-occurrence based stemmer (Paik et al. 2011b)
HPS	High precision stemmer (Brychcín and Konopík 2015)

Table 7 Statistics of test collections used in analysis

Language	Source	No. of documents	No. of queries
English (news collection)	TREC disks 4 and 5	472,525	150 (301–450)
English (web collection)	TREC GOV2	25,205,179	150 (701–850)
Hungarian	CLEF 2006, 2007	49,530	98 (CLEF 2006–2007)
Czech	CLEF 2007	81,725	150 (CLEF 2007)
Bengali	FIRE 2010	123,047	50 (FIRE 2010)
Marathi	FIRE 2008	99,362	50 (FIRE 2008)

news documents from Foreign Broadcast Information Service (FBIS), Financial Times and Los Angeles Times (TREC disk 4 and 5) and topics are taken from TREC 6-7-8 (301–450). The second English experiments are performed on TREC web collection (GOV2) containing nearly 25 million web documents collected from .gov websites with 150 topics (701–850). For Hungarian, document collection and query sets are considered from CLEF 2006 and CLEF 20007 and for Czech the test collection is taken from CLEF 2007. For Asian languages, Bengali and Marathi, documents and query sets are taken from FIRE 2010 and FIRE 2008 respectively.

5.2.2 English results

The retrieval results for various stemmers tested on English (news collection) for TREC 6-7-8 topics (Title and Description) are listed in Table 8. The results show that all the techniques perform better than no stemming, but the difference in MAP is quite small (3–18 %). This confirms the fact that morphological complexity of English is comparatively less than other languages. LING stemmer performed relatively poorer on the English news collection whereas GRAS achieved the highest improvement in MAP. XU, OARD, HMM, and YASS performed almost equally and showed an improvement of about 9 % over unstemmed words. PORT, SNS, HPS also did equally well but are nearly 4 % inferior to GRAS. The query-wise performance of GRAS is statistically compared with all other stemming methods under analysis. We can infer from the *p* values (Table 9) that the performance differences of GRAS are statistically significant over all other methods except HPS.

Table 10 shows retrieval results of various stemmers tested on English web collection with 150 queries (Title field only). In the case of web collection, the performance improvement due to stemming is comparatively less than news collection (2.6–8.6 %). The rule based stemmer, LOVIN showed least improvement of 2.6 % in MAP as compared to no stemming. Interestingly, XU stemmer performed best among all the six stemmers and showed an increase of 8.6 % in MAP as compared to no stemming. PORT, SNS, and GRAS performed almost equally well but are found to be nearly 3 % inferior to XU. But the performance differences of XU with PORT, GRAS, and HPS are not found to be statistically significant as shown in Table 11. The comparison of MAP values among different stemmers tested on English news collection and web collection are shown in Figs. 2 and 3 respectively.

5.2.3 Hungarian and Czech results (European languages)

Tables 12 and 13 list the retrieval performance of various stemmers tested on Hungarian test collection (CLEF 2006–2007) and Czech test collection (CLEF 2007) respectively. Hungarian being a language with high richness and morphological complexity, showed significant improvement in MAP over unstemmed words ranging from 10 to 50 %. Rule-based Snowball stemmer, RB-HU, achieved a consistent improvement of 30 %. Among the lexicon based techniques, OARD, LING, 4-gram, YASS showed comparatively less improvement of 10–28 % in MAP. FCS and GRAS showed a significant improvement of 42.7 and 46.8 % respectively. Among corpus analysis techniques, both HPS and SNS achieved high improvement of 44.7 and 50.2 % respectively over no stemming.

In Czech, stemming is found to be even more beneficial than other languages as the MAP is improved by more than 50 % over unstemmed words. LING, unlike English, performed better on Czech data set with an improvement of nearly 30 %. XU, OARD, and 4-gram techniques performed almost equal with an improvement of nearly 20 % in MAP. YASS and rule-based stemmer (RB-CZ) achieved significant improvements of nearly 44 % over no stemming. GRAS, HPS, and SNS, however, achieved large and significant improvements of more than 50 % in MAP over no stemming. Figures 4 and 5 show the comparison of MAP of various stemming techniques analyzed on Hungarian and Czech collections respectively.

Table 14 shows the statistical comparison of the query-wise performance of average precision values of GRAS with other stemming techniques for both Hungarian and Czech. We can infer from the table that GRAS performs significantly better than rule-based stemmers, XU, OARD, LING, 4-gram and YASS whereas the performance differences are insignificant with FCS, SNS, and HPS for both the languages.

Table 8 Retrieval results for English news collection (TREC disks 4 and 5)

	NO	LOVIN	PORT	XU	OARD	LING	HMM	YASS	GRAS	SNS	HPS
MAP	0.229	0.245 (6.9)	0.260 (13.5)	0.250 (9.1)	0.251 (9.6)	0.236 (3.0)	0.249 (8.7)	0.250 (9.1)	0.270 (17.9)	0.259 (13.1)	0.261 (13.9)
R-P	0.273	0.287 (5.1)	0.301 (10.2)	0.293 (7.3)	0.292 (6.9)	0.280 (2.6)	0.289 (5.8)	0.291 (6.6)	0.309 (13.1)	0.300 (9.8)	0.304 (11.4)
P@10	0.433	0.453 (4.6)	0.483 (11.5)	0.460 (6.2)	0.451 (4.2)	0.449 (3.7)	0.454 (4.8)	0.463 (6.9)	0.479 (10.6)	0.473 (9.2)	0.469 (8.3)
Rel. Ret.	6812	7360 (8.0)	7751 (13.8)	7751 (13.8)	7388 (8.5)	7236 (6.2)	7412 (8.8)	7652 (12.3)	7873 (15.6)	7638 (12.1)	7612 (11.7)

The values in brackets are relative improvements compared with no stemming

Table 9 Statistical significance test results of various stemmers as compared to GRAS for English news collection

Technique	LOVIN	PORT	XU	OARD	LING	HMM	YASS	SNS	HPS
p values	$\mathbf{6.14 \times 10^{-6}}$	**0.036**	**0.002**	**0.001**	$\mathbf{5.85 \times 10^{-7}}$	**0.004**	**0.001**	**0.045**	0.4364

Bold value indicates statistically better performance

Table 10 Retrieval results for English web collection (TREC GOV2)

Technique	NO	LOVIN	PORT	XU	YASS	GRAS	SNS
MAP	0.268	0.275 (2.6)	0.281 (4.9)	0.291 (8.6)	0.278 (3.7)	0.283 (5.6)	0.284 (6.0)
R-P	0.326	0.331 (1.5)	0.335 (2.8)	0.344 (5.5)	0.334 (2.5)	0.335 (2.8)	0.337 (3.4)
P@10	0.540	0.517 (−4.3)	0.522 (−3.3)	0.531 (−1.7)	0.520 (−3.7)	0.541 (0.2)	0.527 (−2.4)
Rel. Ret.	16713	17102 (2.3)	17563 (5.1)	18036 (7.9)	17610 (5.4)	17823 (6.6)	17823 (6.6)

The values in brackets are relative improvements compared with no stemming

Table 11 Statistical significance test results of various stemmers as compared to XU stemmer for English web collection

Technique	LOVIN	PORT	YASS	GRAS	SNS
p values	**0.0123**	0.3870	**0.0201**	0.4267	0.4010

Bold value indicates statistically better performance

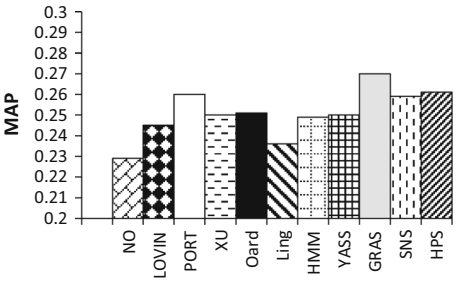

Fig. 2 MAP values for English news collection (TREC disks 4 and 5)

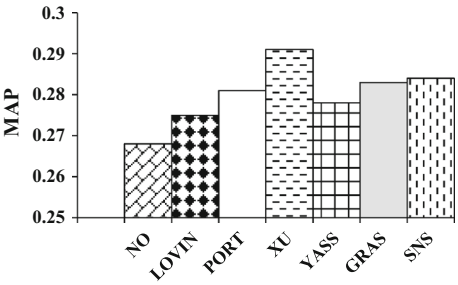

Fig. 3 MAP values for English web collection (TREC GOV2)

Table 12 Retrieval results for Hungarian (CLEF 2006–2007 collection)

Technique	NO	RB-HU	XU	OARD	LING	4-gram	YASS	GRAS	FCS	SNS	HPS
MAP	0.239	0.313 (30.9)	0.285 (19.2)	0.267 (11.7)	0.295 (23.4)	0.271 (13.4)	0.306 (28.0)	0.351 (46.8)	0.341 (42.7)	0.359 (50.2)	0.346 (44.7)
R-P	0.252	0.312 (23.8)	0.302 (19.8)	0.285 (13.1)	0.310 (23.0)	0.295 (17.1)	0.315 (25.0)	0.360 (42.8)	0.352 (39.7)	0.358 (42.0)	0.351 (39.2)
P@10	0.314	0.399 (27.1)	0.339 (7.9)	0.340 (8.3)	0.358 (14.0)	0.349 (11.1)	0.384 (22.3)	0.422 (34.4)	0.390 (24.2)	0.422 (34.4)	0.417 (32.8)
Rel. Ret	1367	1723 (26.0)	1803 (31.8)	1546 (13.1)	1738 (27.1)	1674 (22.5)	1730 (26.6)	1924 (40.7)	1912 (39.9)	1964 (43.6)	1917 (40.2)

The values in brackets are relative improvements compared with no stemming

Table 13 Retrieval results for Czech (CLEF 2007 collection)

Technique	NO	RB-CZ	XU	OARD	LING	4-gram	YASS	GRAS	FCS	SNS	HPS
MAP	0.238	0.341 (43.2)	0.286 (20.1)	0.277 (16.4)	0.308 (29.4)	0.280 (17.6)	0.342 (43.7)	0.366 (53.8)	0.351 (47.4)	0.363 (52.5)	0.357 (50.0)
R-P	0.261	0.346 (32.6)	0.289 (10.7)	0.286 (9.6)	0.307 (17.6)	0.286 (9.6)	0.329 (26.1)	0.360 (37.9)	0.336 (28.7)	0.344 (31.8)	0.358 (37.1)
P@10	0.268	0.348 (29.9)	0.304 (13.4)	0.298 (11.2)	0.336 (25.4)	0.301 (12.3)	0.340 (26.9)	0.376 (40.3)	0.356 (32.8)	0.370 (38.1)	0.380 (41.7)
Rel. Ret	551	667 (21.1)	632 (14.7)	615 (11.6)	664 (20.5)	620 (12.5)	672 (22.0)	684 (24.1)	672 (22.0)	681 (23.6)	680 (23.4)

The values in brackets are relative improvements compared with no stemming

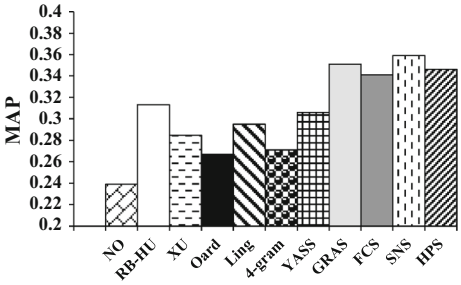

Fig. 4 MAP values for Hungarian (CLEF 2006–2007)

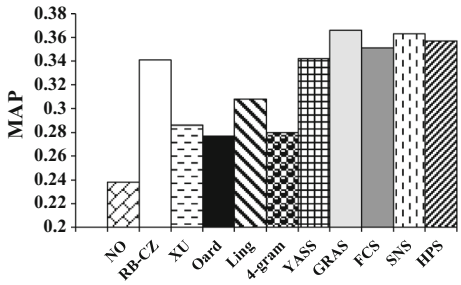

Fig. 5 MAP values for Czech (CLEF 2007)

Table 14 Statistical significance test results of various stemmers as compared to GRAS for Hungarian and Czech

Language	Rule	Xu	OARD	LING	4-gram	YASS	FCS	SNS	HPS
Hungarian	**0.0003**	**0.34E−6**	**1.65E−8**	**0.75E−5**	**2.12E−7**	**1.48E−5**	0.2218	0.5567	0.4324
Czech	**0.048**	**2.54E−4**	**3.40E−5**	**0.0004**	**5.67E−5**	**0.0162**	0.1278	0.6314	0.3468

Bold value indicates statistically better performance

5.2.4 Bengali and Marathi results (Asian languages)

The retrieval performance of various stemmers on two Asian languages namely Marathi (FIRE 2010 collections) and Bengali (FIRE 2008 collections) is presented in Tables 15 and 16. Like Hungarian, Marathi is also highly agglutinative and inflectional in nature and hence stemming is found to be very beneficial for this language. Marathi rule based stemmer (RB-MA) and OARD performed worse on Marathi collections and showed a very little improvement of 3–4 % only. YASS, HPS, and LING performed fairly well in terms of both MAP and relevant documents retrieved. GRAS and SNS are the top performers for Marathi wherein the improvement in MAP is up to 43 % as compared to unstemmed words.

Table 16 reports the retrieval performance of stemmers tested on Bengali test collection (FIRE 2008). Unlike other European and Asian languages discussed in the paper, stemming is found to be less advantageous on Bengali test collection as it resulted in maximum improvement of 21 % in MAP over unstemmed words. OARD and LING reported small improvements of only 6.6 and 8.6 % whereas rule-based stemmer performed slightly better than them. YASS and HPS showed a significant improvement of nearly 18 %. GRAS and SNS performed almost equal. Figures 6 and 7 show the comparison of MAP of various stemming techniques analyzed on Marathi and Bengali collections respectively.

Table 17 reports statistical results for the query-wise performance of GRAS with other stemming methods for both Marathi and Bengali. It is clear from there that GRAS performs statistically better than all the stemmers (except SNS) in the Marathi language. In the case of Bengali, the performance of GRAS is significantly better than rule-based, OARD and LING stemmers but the performance difference of YASS, SNS, and HPS with GRAS are statistically insignificant.

5.2.5 Analysis of results

In this section, we present our findings and discuss some merits and demerits of the stemmers on the basis of the analysis of retrieval results. Though the results presented above cannot be used for comparing different stemming techniques as they use different data set, yet they provide a useful insight of the performance of various stemmers.

Among the rule-based stemmers for English, Lovin did fairly well whereas the performance of Porter stemmer is better or at par with most of the language-independent stemmers except for GRAS. One of the limitations of Porter stemmer in retrieval tasks is that it does not remove the possessive markers from the word endings. So, words like *strongest* are not stemmed to *strong*. For European languages, Czech and Hungarian, the rule-based stemmers (RB-CZ and RB-HU) performed moderately well in retrieval tests. But for Asian languages, Marathi and Bengali, both rule-based stemmers (RB-MA and RB-BE) performed worse than most of the statistical stemmers in the retrieval experiments. The moderate or poor performance of European and Asian rule based stemmers is probably due to use of light stemming procedures that consider only the inflectional variations caused due to nouns and adjectives and thereby ignoring various changes due to verb forms or frequently occurring derivational suffixes.

In the category of lexicon analysis based statistical stemmers; GRAS is the top performer in all the languages in terms of computational efficiency, improving mean average precision, and relevant documents retrieved. Fast corpus-based stemmer (FCS) also enhances retrieval efficiency and is found to be quite close to GRAS in some languages. In some cases, the performance of FCS is degraded as it ignores the infrequent suffixes (suffixes with frequency below the desired cut-off). So, it sometimes separates a valid class into a number of equivalence classes. YASS, another lexicon based stemmer is found to be highly aggressive as it groups the largest number of words in each conflation class for all the languages. YASS performed fairly well in English, Bengali, and Czech, but comparatively less effective on highly inflectional languages like Hungarian and Marathi. This is due to the reason that complete linkage clustering technique cannot handle large equivalence classes.

4-Gram and HMM techniques also showed consistent performance in handling morphological changes in the alphabetic languages. In comparison with other statistical techniques discussed in this section, n-gram technique can not only handle inflectional variations in the suffixing languages but can also handle changes due to derivational variations, compounding of words or spelling variations. LING is found to be less effective than other stemmers in the tests, particularly in English and Bengali test collections. Moreover, LING seems to be computationally intensive, with the time taken for implementation of initial tasks for an average sized test collection running into days. On the other hand, OARD stemmer is found to be computationally simplest which performs light stemming through suffixes discovered from the corpus. Due to which it is unable to handle the complex and rich morphology of Marathi and Hungarian and hence did not show consistent improvement in European and Asian languages.

Table 15 Retrieval results for Marathi (FIRE 2010 collection)

Techniques	NO	RB-MA	OARD	LING	YASS	GRAS	SNS	HPS
MAP	0.315	0.329 (4.4)	0.327 (3.8)	0.411 (30.5)	0.372 (18.1)	0.451 (43.2)	0.437 (38.7)	0.375 (19.0)
R-P	0.315	0.313 (−0.6)	0.330 (4.7)	0.393 (24.8)	0.347 (10.2)	0.430 (36.5)	0.418 (32.7)	0.377 (19.7)
P@10	0.356	0.367 (3.0)	0.336 (−5.6)	0.380 (6.7)	0.377 (5.9)	0.380 (6.7)	0.388 (8.9)	0.374 (5.1)
Rel. Ret	551	555 (0.7)	561 (1.8)	590 (7.1)	589 (6.9)	607 (10.2)	602 (9.3)	585 (6.1)

The values in brackets are relative improvements compared with no stemming

Table 16 Retrieval results for Bengali (FIRE 2008 collection)

Techniques	NO	RB-BE	OARD	LING	YASS	GRAS	SNS	HPS
MAP	0.393	0.443 (12.7)	0.427 (8.6)	0.419 (6.6)	0.463 (17.9)	0.476 (21.1)	0.474 (20.6)	0.466 (18.6)
R-P	0.367	0.410 (11.7)	0.406 (10.6)	0.369 (0.5)	0.415 (13.0)	0.435 (18.5)	0.428 (16.6)	0.415 (13.0)
P@10	0.352	0.362 (2.8)	0.344 (−2.3)	0.334 (−5.1)	0.372 (5.7)	0.370 (5.1)	0.369 (4.8)	0.358 (1.7)
Rel. Ret	489	499 (2.0)	495 (1.2)	499 (2.0)	498 (1.8)	499 (2.0)	500 (2.2)	499 (2.0)

The values in brackets are relative improvements compared with no stemming

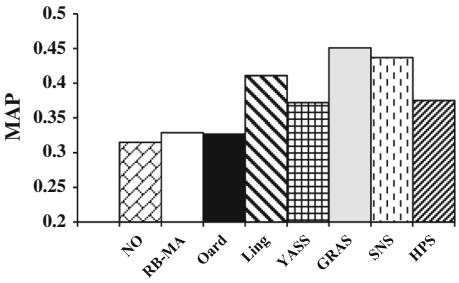

Fig. 6 MAP values for Marathi (FIRE 2010)

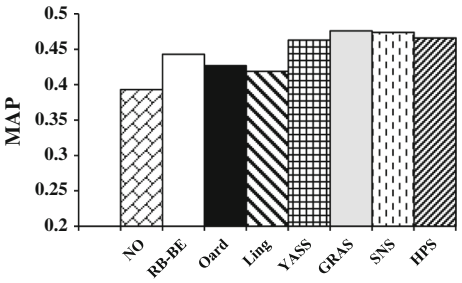

Fig. 7 MAP values for Bengali (FIRE 2008)

Table 17 Statistical significance test results of various stemmers as compared to GRAS for Marathi and Bengali

Language	RULE	OARD	LING	YASS	SNS	HPS
Marathi	**0.003**	**0.0005**	**0.040**	**0.015**	0.128	**0.021**
Bengali	**0.041**	**0.0360**	**0.008**	0.140	0.458	0.238

Bold value indicates statistically better performance

Corpus analysis techniques show certain features of the underlying corpus and are found to be more effective as compared to rule-based stemmers. Moreover, the performance of corpus analysis based stemmers can further be enhanced by using background stemmers. But, corpus analysis based techniques require relatively larger corpus so as to enhance the dependability of co-occurrence information. Among the corpus analysis based stemmers; XU that uses co-occurrence data gave an average performance in all the languages because the graph clustering algorithms carry some weaknesses. The connected component technique creates long morphological classes, and optimal partitioning algorithm requires background stemmer to handle the complex languages. SNS, that is also developed on the similar line as XU but uses simple co-occurrence measure performed consistently well in all the languages. The high performance of SNS is the result of the coupling of the nearest neighbor algorithm with the recalculated co-occurrence weights. HPS performed fairly well in all the languages except Marathi. Moreover, it required very small training data as compared to other stemmers of the same category and showed a high out-of-vocabulary rate.

On the basis of our analysis, we suggest the best stemmers for retrieval tasks in each category. One can select from these stemmers according to the availability of resources for training the stemmers. For instance, if a user has relatively small document collection or only lexicon of a particular language then lexicon analysis based stemmer can be chosen. Similarly, if one has large document collection, then corpus analysis based stemmer can be chosen. Among the seven lexicon analysis based stemmers analyzed in this section, we

found GRAS is most suitable for retrieval tasks in all the languages. It is fast, aggressive and gives the best retrieval results in all the language families. After GRAS, FCS and YASS also perform consistently well and can be used effectively in retrieval experiments. YASS is not suitable for highly inflectional languages like Marathi and Hungarian as it cannot handle large equivalence classes due to a large number of suffixes. Among corpus analysis based stemmers, HPS and SNS are most suitable in retrieval experiments for all the language families. The major advantage of HPS is that it requires small training data and can be trained even on a small subset of whole test collection. Moreover, because of the significant out-of-vocabulary rate it can be used to index unseen document collection. SNS also forms accurate equivalence classes using relatively simpler co-occurrence measure and a novel nearest neighbor based clustering algorithm. Among rule based stemmers, Porter is most suitable for English and can be used in retrieval experiments. The rule-based stemmers belonging to Asian languages should not be preferred in retrieval tasks as they cannot handle a large number of inflectional variations (such as variations in verb forms or frequently occurring derivational suffixes) and perform light stemming.

6 Factors affecting stemmer performance

The performance of the stemmer varies according to a number of factors such as the size of the corpus, nature of documents, type of task, etc. In this section, we intend to study the dependence of stemmer effectiveness on various factors through series of experiments. On the basis of evaluation results presented in Sect. 5.2, we chose six stemmers, two each from language-specific, lexicon analysis based and corpus analysis based category. Among language-specific stemmers, we considered Porter stemmer (Porter 1980) and Lovins stemmer (Lovins 1968) while from lexicon analysis based category graph based stemmer (GRAS) (Paik et al. 2011a), yet another suffix stripper (YASS) (Majumder et al. 2007b) are considered. Among corpus analysis category, we considered co-occurrence statistic based word clustering (SNS) (Paik and Parui 2011) and high precision stemmer (HPS) (Brychcín and Konopík 2015).

The evaluations reported in the following subsections are based on TREC FBIS, Financial Times and LA Times document collection (TREC disks 4 and 5) containing 4,72,525 articles. The retrieval experiments are carried out through title and description fields of TREC 6-7-8 topics (301–450) on TERRIER IR system. In all the experiments, mean average precision (MAP), recall-precision (R-P), Precision@10 (P@10) and relevant documents retrieved (Rel. Ret.) measures are reported. In the first four subsections, retrieval performance of stemmers is analyzed according to different factors and in the last subsection, stemmer performance is evaluated in two different scenarios i.e. web search and text classification.

6.1 Information retrieval models

The study of stemming algorithms according to different retrieval models is central to information retrieval domain. We evaluated various stemmers with respect to different indexing and searching strategies. To analyze various stemming techniques on different IR models, we used five models from different paradigms. First, we used well known Okapi BestMatch25 (BM25) model (Robertson et al. 2000). Okapi BM25 is a classical *probabilistic model,* used for large text collection which incorporates document length, document term frequency and query term frequency for improving idf term in the weighting function. From the *Divergence of Randomness* (DFR) paradigm, IFB2, PL2 (Amati and Rijsbergen 2002) and DLH (Amati

Table 18 Retrieval results of various stemming techniques and IR models

Parameters	NO STEM	Lovin	Porter	YASS	GRAS	SNS	HPS
Okapi BM25 model							
MAP	0.2158	0.2391	0.2508	0.2431	0.2585	0.2510	0.2481
R-Precision	0.2662	0.2833	0.2968	0.2841	0.2960	0.2912	0.2962
P@10	0.4233	0.4480	0.4680	0.4568	0.4667	0.4612	0.4600
Rel. Ret	6636	7184	7619	7486	7683	7478	7443
PL2 model							
MAP	0.2059	0.2237	0.2383	0.2282	0.2434	0.2356	0.2345
R-Precision	0.2535	0.2716	0.2879	0.2740	0.2881	0.2851	0.2850
P@10	0.4133	0.4393	0.4567	0.4463	0.4553	0.4503	0.4507
Rel. Ret	6446	6929	7362	7285	7422	7263	7217
Hiemstra language model							
MAP	0.1931	0.2102	0.2293	0.2074	0.2283	0.2214	0.2238
R-Precision	0.2367	0.2519	0.2719	0.2515	0.2670	0.2581	0.2737
P@10	0.3560	0.3687	0.4013	0.3782	0.3912	0.3923	0.3880
Rel. Ret	6040	6604	6993	6840	7037	6799	6767
DLH model							
MAP	0.2233	0.2378	0.2541	0.2411	0.2588	0.2527	0.2532
R-Precision	0.2719	0.2828	0.2997	0.2883	0.3006	0.2982	0.2991
P@10	0.4367	0.4567	0.4693	0.4567	0.4685	0.4619	0.4573
Rel. Ret	6609	7059	7504	7393	7578	7367	7388
IFB2 model							
MAP	**0.2289**	**0.2449**	**0.2596**	**0.2499**	**0.2698**	**0.2582**	**0.2608**
R-Precision	0.2736	0.2868	0.3013	0.2917	0.3090	0.3001	0.3036
P@10	0.4320	0.4553	0.4827	0.4634	0.4792	0.4727	0.4693
Rel. Ret	6812	7360	7760	7652	7873	7638	7612

Bold value indicates the best result in each column

2006) models are included. IFB2 model uses inverse term frequency as basic randomness model with Bernoulli after-effect as first normalization and normalization2 as term frequency normalization. PL2 model is another well-known model from the DFR paradigm which is used for applications which need early precision. PL2 uses Poisson model as basic randomness model with Laplace after-effect normalization and normalization2 as term frequency normalization.

Further, a parameter free DLH model is used for analysis from the DFR paradigm. DLH model is based on hyper-geometric term frequency distribution which is reduced to binomial distribution based on non-uniform term prior distribution for the sake of obtaining a viable weighting function. Finally, we used Hiemstra model (Hiemstra 2001) from the language model (LM) paradigm. Hiemstra model uses Jelinek–Mercer smoothing and combines approximation based on entire corpus and documents. The evaluation results of stemmers with respect to five different information retrieval models are presented in Table 18.

In Table 18, the best MAP values for all the stemming techniques are marked in bold, indicating that the IFB2 model from the DFR paradigm always happens to be the best IR model. As we can see from Fig. 8, the performance of DLH model and Okapi BM25 model in the case of different stemming techniques is almost equal but inferior to IFB2. The perfor-

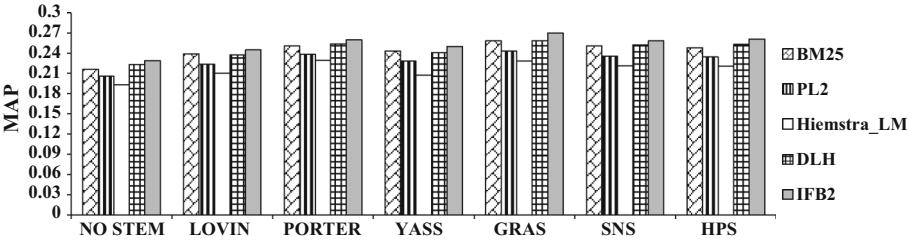

Fig. 8 MAP values for different stemming techniques with respect to various IR models

Table 19 Statistical significance tests results of various stemmers w.r.t. different IR models as compared to IFB2 model

IR models	LOVIN	PORTER	YASS	GRAS	SNS	HPS
Okapi BM25	0.1234	**0.0009**	**0.0010**	**2.03E−5**	**1.21E−4**	**3.91E−8**
Hiemstra_LM	**1.05E−5**	**0.0002**	**3.0E−8**	**6.12E−6**	**0.0004**	**0.0001**
PL2	**5.40E−5**	**7.62E−6**	**8.60E−7**	**1.69E−7**	**1.11E−6**	**4.44E−8**
DLH	0.1466	0.2052	**0.0291**	**0.0068**	0.1217	0.1399

Bold value indicates statistically better performance

mance of PL2 model from DFR paradigm was found to be less than other models of the same paradigm and classical BM25 model. Hiemstra model from the language model paradigm performed the worse.

We considered IFB2 model as the baseline for our statistical testing. The query-wise performance of each stemming technique using IFB2 model is compared with all other models. The p values using paired t tests are reported in Table 19. We can thus see that the performance of IFB2 model is statistically better than PL2, Hiemstra, and Okapi BM25 model (except for Lovins stemmer with BM25 indexing scheme). The performance difference between IFB2 and DLH are found to be statistically insignificant except for the stemmers belonging to the lexicon analysis based category.

Thus, we conclude that the IFB2 model from the DFR paradigm provides the best retrieval results in all the stemming approaches as compared to other IR models of the same paradigm or other paradigms. So, in rest of the analysis presented in the subsequent subsections, we adopted IFB2 model as information retrieval model.

6.2 Size of corpus

The size of training data is another important factor that needs to be analyzed while comparing the performance of different stemmers. Statistical stemmers are often computationally expensive as they involve a lot of calculations and procedures like clustering, graph related activities, etc. An appropriate size of the corpus would decrease the time and amount of computations involved in the stemmer development process. In order to analyze this factor, we conducted three different runs by providing different size of training data (subsets of corpora). The complete statistics of the training data according to all the three runs is given in Table 20. In the first run, all the stemmers are trained on documents of Foreign Broadcast Information Service (FBIS) from TREC disks 4 and 5. In the second run, the size of training data is increased by incorporating documents of Financial Times (FT) along with FBIS from the same source. In the third run, the entire corpus is considered which includes documents of FBIS, FT and LA Times.

Table 20 Statistics of training data according to different size

Corpus	Source	Size (in MB)	No. of files	No. of documents
FBIS	TREC disks 4 and 5	493	492	130,471
FBIS + FT	TREC disks 4 and 5	1084.5	1085	340,629
FBIS + FT + LA Times	TREC disks 4 and 5	1582.9	1815	472,525

By analyzing the retrieval results of all the three runs from Table 21, we infer that both the rule-based stemmers (Lovin and Porter) and lexicon analysis based stemmers (YASS and GRAS), are not much affected by the size of corpus used in training. The MAP values in both categories in all the three runs are nearly same, and there is a small change in the total number of relevant documents retrieved. This is clear from Figs. 9 and 10, where the comparison between the MAP values and Relevant Documents retrieved with the size of training data is shown. The minor differences in MAP values in the case of rule-based and lexicon analysis based stemmers is probably because these stemmers do not learn any statistics about the words from the surrounding contexts in the corpus. The subset of the corpus is sufficient for these stemmers to give consistent performance.

In case of corpus analysis based stemmer (SNS), there is a considerable increase in MAP values and number of relevant documents retrieved with an increase in corpus size. This is due to the reason that the SNS stemmer learns the entire information about the words from the co-occurrence statistics which is dependent on the size of the corpus. With the increase in the size of the corpus, term frequency of the words increase and hence more accurate equivalence classes are obtained. High precision stemmer (HPS), on the other hand, is less affected by the size of corpus both in terms of MAP and relevant documents retrieved. This is because HPS is not totally dependent upon the corpus statistics; rather it also considers other lexical features like suffix, n-gram, word length information and lexical similarity.

We considered results of run III as a baseline for our statistical testing and compared the query wise average precision values using paired t test. This is clear from the p values shown in Table 22, that the difference between different runs in case of rule-based, lexicon analysis based and HPS is statistically insignificant whereas in case of SNS which is purely corpus analysis based technique the difference is statistically significant when the size of corpus is approximately one-third of the whole corpus (Run-I).

Thus, we conclude that stemmers belonging to purely corpus analysis based category such as SNS are affected by the size of the corpus. The MAP and total relevant documents increase as we keep on increasing the size of the training data. The rule-based and lexicon analysis based stemmers are not much affected by the size of the corpus both in terms of MAP and relevant documents retrieved.

6.3 Type of training

The performance of stemmers is usually tested by training the stemmer on the same data collection that is to be indexed. In real situations, the data is continuously increasing, and new data emerge frequently. The stemmers thus need to be retrained to index these new test collections. Retraining of stemmers is usually computationally expensive. In such situations, those stemmers are preferred which can handle unseen data efficiently and avoid the need for retraining. So, we decided to test the retrieval performance of stemmers by training them on both types of data:

Table 21 Retrieval results of various stemming techniques according to different size of training data

	Run I: stemmers trained on FBIS documents (one-third collection)				Run II: stemmers trained on FBIS and FT documents (two-third collection)				Run III: stemmers trained on FBIS, FT, LA Times documents (full collection)			
	MAP	R-P	P@10	Rel. Ret.	MAP	R-P	P@10	Rel. Ret.	MAP	R-P	P@10	Rel. Ret.
LOVIN	0.2443	0.2868	0.4553	7338	0.2444	0.2870	0.4553	7355	**0.2449**	0.2868	0.4553	**7360**
PORTER	0.2594	0.3011	0.4820	7745	0.2596	0.3016	0.4814	7756	**0.2596**	0.3013	0.4827	**7760**
YASS	0.2487	0.2939	0.4593	7644	0.2455	0.2937	0.4607	7650	**0.2499**	0.2917	0.4634	**7652**
GRAS	0.2692	0.2999	0.4740	7812	0.2697	0.3003	0.4753	7880	**0.2698**	0.3090	0.4792	**7873**
SNS	0.2457	0.2912	0.4642	7436	0.2562	0.2967	0.4700	7590	**0.2582**	0.3001	0.4727	**7638**
HPS	0.2592	0.3050	0.4673	7569	0.2605	0.3062	0.4713	**7629**	**0.2608**	0.3036	0.4693	7612

Bold value indicates the best results of MAP and Rel. Ret.

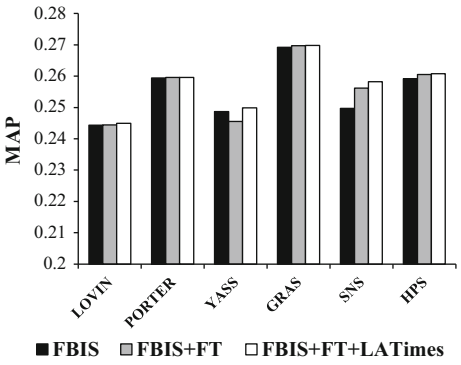

Fig. 9 MAP values for various stemmers wrt different corpus size

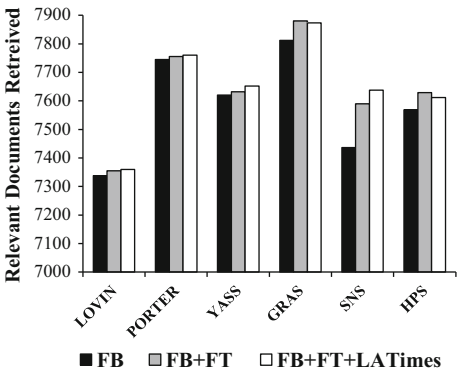

Fig. 10 Relevant documents retrieved for various stemmers wrt different corpus size

Unseen: In this case, the data used during training and indexing of documents is different. The words which are to be indexed are therefore not known by the stemmers. We trained the stemmers on the documents of Wall Street Journal (WSJ), Associated Press (AP) and Information from computer select disks (Ziff) provided by TREC disks 1 and 2 whereas the stemmers are used to index the documents of Foreign Broadcast Information Service (FBIS), Financial Times (FT) and Los Angeles Times (LA Times) provided by TREC disks 4 and 5.

Seen: In this case, stemmers are trained on the same data on which the indexing is to be performed. So, all the words to be indexed are already known to the stemmers. For performing these experiments, we trained and tested the stemmers on documents of FBIS, FT and LA Times. The complete statistics of documents is provided in Table 23. To avoid any bias, the size of training and testing data is nearly same in both the cases.

The retrieval results for both seen and unseen training of stemmers are presented in Table 24. We can thus see from the results that in all the stemmers the difference in MAP and number of relevant documents retrieved is very small. Interestingly, in the case of PORTER and YASS stemmers, the MAP for unseen training is slightly more than that of seen training. The query-wise performance in case of each stemmer for both seen and unseen training has been statistically tested using paired t test. The p values in Table 25 confirm that the difference in the retrieval results of both types of training is statistically insignificant for all the stemmers under analysis.

Table 22 Statistical significance test results of runs I and II as compared to run III

	LOVIN	PORTER	YASS	GRAS	SNS	HPS
FBIS (run-I)	0.8633	0.3918	0.8186	0.8680	**0.0009**	0.6592
FBIS + FT (run-II)	0.4860	0.3180	0.3085	0.3182	0.3074	0.9900

Bold value indicates statistically better performance

Table 23 Statistics of test collections used for seen and unseen training

Type of training	Training corpus	Source	Size (in GB)	No. of documents	No. of words
Unseen	WSJ, AP, ZIFF	TREC disks 1 and 2	1.5	469,949	518,471
Seen	FBIS, FT, LA Times	TREC disks 4 and 5	1.58	472,525	522,381

Table 24 Retrieval results for seen and unseen training

	Unseen: training and testing data are different				Seen: training and testing data are same			
	MAP	R-P	P@10	Rel. Ret	MAP	R-P	P@10	Rel. Ret
LOVIN	0.2448	0.2871	0.4567	7355	0.2449	0.2868	0.4553	7360
PORTER	0.2617	0.3034	0.4813	7765	0.2596	0.3013	0.4827	7760
YASS	0.2535	0.2995	0.4639	7672	0.2499	0.2917	0.4634	7652
GRAS	0.2690	0.3031	0.4713	7743	0.2698	0.3090	0.4792	7873
SNS	0.2571	0.2999	0.4707	7590	0.2582	0.3001	0.4727	7638
HPS	0.2580	0.3037	0.4693	7550	0.2608	0.3036	0.4693	7612

Table 25 Statistical significance tests for seen and unseen training (150 observations)

Stemmers	LOVIN	PORTER	YASS	GRAS	SNS	HPS
p values	0.7213	0.6022	0.4470	0.9217	0.5658	0.4335

6.4 Nature of documents

Another important factor that may affect the performance of stemmers is the type of documents used during training. In all the experiments reported in the above subsections, stemmers are trained on news collections. In this subsection, we evaluated the retrieval performance of stemmers by training them on different types of documents. We performed three retrieval experiments using different types of training data in each case. Firstly, we trained the stemmers on collection provided by Open American National Corpus (OANC) (available at http://www.anc.org/data/oanc). OANC provides open text collection from different domains such as web collection, fiction and non-fiction articles, travel guides, letters, technical and journal articles. In the second run, we trained the stemmers on San Jose Mercury News articles provided by Text Retrieval Conference Collection. In the third run, stemmers are trained on a combination of articles from OANC and SJM news. In all the three experiments, the size of the text collections is approximately the same so as to avoid any bias. Moreover, in all

the three cases, unseen training is provided to the stemmers i.e. the training data and text collections to be indexed are different.

The retrieval results of all the three runs are shown in Table 26. As we can see from the table that in the case of rule-based stemmers (LOVIN and PORTER) the change in MAP and number of relevant documents retrieved with different types of training documents is very small. This indicates that rule based stemmers are not much affected by the nature of documents used during training. In the category of lexicon analysis based stemmers (YASS and GRAS), there is a significant increase in MAP and number of relevant documents retrieved. In the case of YASS, MAP increased by 5.5 % when a combination of web, news, technical, fiction and non-fiction articles are used as compared to only web collection or only news collection. In the case of GRAS, MAP increased by 6.3 % and 189 more relevant documents are retrieved when a combination of different nature of documents are used as compared to the collection of OANC only.

In the category of corpus analysis based statistical stemmers also, there is a significant increase in MAP and number of relevant documents retrieved. In the case of SNS, MAP increased by 4.7 % and 176 more relevant documents are retrieved when a combination of training data is used while in the case of HPS there is a comparatively small increase in MAP (2.2 %) but the number of relevant documents increased by 152 when a combination of all types of documents is used for training. The comparison between MAP and number of relevant documents retrieved in case of all the stemmers for all the three runs are shown in Figs. 11 and 12 respectively.

The query wise average precision values of run three (combination of text from OANC and SJM) are statistically compared with results from run one and two using paired t test. The p values in Table 27 confirms that rule based stemmers are not affected by nature of documents used in training whereas in the case of statistical stemmers (YASS, GRAS, SNS) the differences in performance are statistically significant.

6.5 Nature of task

In all the above experiments presented in this section, the evaluation of stemmers is done through information retrieval tasks. In this subsection, we applied various linguistic and statistical stemmers on two different scenarios: *web search and text classification*. The purpose is to reveal stemmer performance according to different type of tasks.

6.5.1 Web search

The web is a quick and direct source to answer questions related to any topic and web search has become a major and important tool in our daily lives for gathering information. Stemming helps the users in making the search process easier as users need not care about the word variants and inflections. With the help of stemming, query expansion mechanism is performed by the system in which query words are stemmed to the common stem (Melucci and Orio 2003). Stemming is important from the viewpoint of search engine optimization, as while the search engine tends to stem the query words still they favor the results that exactly match the original query words. So, we intend to study the effect of various stemming methods on the web search task.

We performed all our searching experiments using Terrier web search tool (http://terrier.org). Terrier web search application provides a web-based front end to perform indexing and searching of websites. It allows the users to crawl new websites, index various documents from the crawled websites, perform searching and write index to the disk. Stemming is per-

Table 26 Retrieval results for stemmers trained on different types of documents

	Run I: stemmers trained on OANC				Run II: stemmers trained on SJM news articles				Run III: stemmers trained on documents from OANC and SJM			
	MAP	R-P	P@10	Rel. Ret.	MAP	R-P	P@10	Rel. Ret.	MAP	R-P	P@10	Rel. Ret.
LOVIN	0.2456	0.2876	0.4567	7353	0.2442	0.2887	0.4560	7355	**0.2467**	0.2869	0.4567	**7357**
PORTER	0.2602	0.2983	0.4793	7752	0.2602	0.2986	0.4807	7762	**0.2618**	0.3004	0.4813	**7765**
YASS	0.2469	0.2932	0.4533	7578	0.2476	0.2913	0.4627	7533	**0.2606**	0.3031	0.4787	**7600**
GRAS	0.2578	0.2952	0.4593	7685	0.2638	0.2957	0.4672	7792	**0.2741**	0.3054	0.4793	**7874**
SNS	0.2483	0.2914	0.4524	7502	0.2499	0.2946	0.4643	7597	**0.2602**	0.2999	0.4718	**7678**
HPS	0.2551	0.2994	0.4620	7445	0.2556	0.3010	0.4647	7451	**0.2609**	0.3052	0.4733	**7597**

Bold value indicates the best results of MAP and Rel. Ret.

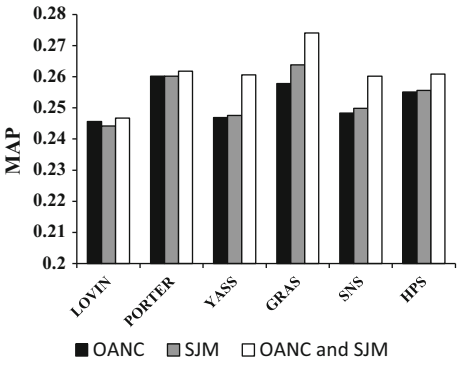

Fig. 11 MAP values for various stemmers according to different nature of documents

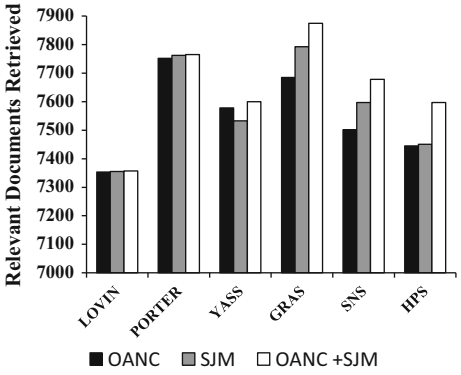

Fig. 12 Relevant documents retrieved for various stemmers according to different nature of documents

Table 27 Statistical significance test results of runs I and II as compared to run III

	LOVIN	PORTER	YASS	GRAS	SNS	HPS
OANC (run-I)	0.7457	0.4204	**0.0252**	**0.0416**	**0.0372**	0.2930
SJM (run-II)	0.4023	0.3549	**0.0165**	**0.0439**	**0.0412**	0.1575

Bold value indicates statistically better performance

formed both at the time of indexing and searching. In order to evaluate the performance of stemming on web search, we manually formulated twenty queries covering various fields such as business, education, science and technology, health, lifestyle, etc. The queries are formulated in such a way that the terms contain various variants and inflections of the original word forms. Some of the examples of queries used are "falling oil prices", "evidences of viruses outbreak", "educational standards", "robotic technology", "relocation of refugees" etc. The web data is collected from various international news websites namely BBC News (http://www.bbc.com), WorldPress (http://www.worldpress.org), Yahoo News (http://www.yahoo.com/news), and CNN (http://edition.cnn.com). The data is collected from news websites because they cover data from the entire world pertaining to a wide range of subjects. The evaluation results are presented in Table 28. We manually evaluated the top-30 documents retrieved for each query and classified them into relevant, irrelevant, partially relevant or duplicate links.

Table 28 Web search performance for top-30 hits for all 20 queries (n = 600)

Technique	Relevant	Irrelevant	Partially relevant	Duplicate links
NO STEM	254 (42.3)	244 (40.6)	68 (11.3)	34 (5.8)
LOVIN	240 (40.0)	208 (34.6)	98 (16.4)	54 (9.0)
PORTER	286 (47.7)	162 (27.0)	132 (22.0)	20 (3.4)
YASS	256 (42.7)	190 (31.7)	92 (15.3)	62 (10.3)
GRAS	264 (44.0)	198 (33.0)	108 (18.0)	30 (5.0)
SNS	236 (39.3)	200 (33.3)	94 (15.7)	70 (11.7)
HPS	260 (43.3)	214 (35.7)	106 (17.6)	20 (3.4)

The values in brackets denote relative percentages with respect to total documents

As we can see from the Table 28, that Porter stemmer showed maximum improvement in web search as compared to no stemming. It retrieves a maximum number of relevant documents (5 % more than no stemming) and a minimum number of irrelevant and duplicate links. The performance of LOVIN and SNS is found to be 2–3 % inferior as compared to no stemming. Statistical stemmers, YASS, GRAS, and HPS, performed slightly better than no stemming by retrieving nearly 20–25 more relevant documents. However, statistical stemmers decreased the number of irrelevant retrievals thereby increasing partially relevant documents. The reason for low-performance improvement of stemmers in web search is that traditional stemmers perform blind stemming i.e. they stem each term of the query without considering the context of the query terms. Stemming should be employed very carefully in web search engines, and one must consider the context of the original query terms while stemming them.

6.5.2 Text classification

Text classification or categorization is the process of assigning predefined labels to the documents. The various documents may belong to a single, multiple or no class at all. With an increasing number of online documents, text classification has become a dominant method to categorize the data. Text classifiers use machine learning techniques to learn from examples and then perform classification automatically. Statistical classification techniques classify the text by estimating the probabilities of the words or n-grams in the documents for any specific class. The estimation of these probabilities is difficult as there are lots of infrequent words in the text collection. Stemming is one way to deal with this problem (Gaustad et al. 2002). Stemming algorithms reduce the number of variant words to a single stem. The frequency of these stems will be more as compared to the frequencies of the inflected word forms. Therefore, the probabilities can be computed more reliably from the collection.

We performed various classification experiments using RTextTools (Jurka et al. 2013) package provided by the R software (available at http://www.R-project.org). The classification experiments are performed on US Congress data collection provided in the R software. US Congress is a sample collection of classified bills from the United State Congress. The classification is performed using support vector machine classifier (SVM) (Meyer et al. 2012). SVMs are universal learners, and they learn through threshold functions. Their learning ability is independent of the size of the feature sets. The major characteristics of SVM classifiers are that they can deal with high dimensional inputs, sparse document vectors and linearly separable categories (Joachims 1998). The document collection is divided into two parts for training and testing, and the same division is used for all the stemmers under analysis. The percentage of documents selected for training is taken to be 60 and rest 40 percent are selected

Table 29 Classification performance of various stemmers using SVM classifier

Technique	Precision	Recall	F-Score
NO STEM	0.6210	0.6290	0.6100
LOVIN	0.6560 (5.6)	0.6535 (3.9)	0.6420 (5.2)
PORTER	0.6830 (10.0)	0.6530 (3.8)	0.6540 (7.2)
YASS	0.6655 (7.2)	0.6540 (4.0)	0.6430 (5.4)
GRAS	0.6770 (9.2)	0.6665 (6.0)	0.6550 (7.4)
SNS	0.6644 (7.0)	0.6540 (4.0)	0.6411 (5.1)
HPS	0.6570 (5.7)	0.6480 (3.0)	0.6385 (4.7)

The values in brackets represent relative improvement as compared to no stemming

for testing. The classification performance of various stemmers is presented in Table 29 in terms of precision, recall, and F-Score. Precision refers to the proportion of documents which the method predicts to be belonging to some class actually belongs to that class. Recall, on the other hand, is the percentage of documents in a class that the classifier correctly assigns to that class. F-Score represents a weighted mean of both precision and recall.

It is clear from the Table 29 that use of stemmer improves the classification accuracy of the classifier on the US Congress text collections as both the precision and recall values for all the stemmers are more than no stemming. The best performance in the classification task is shown by Porter and GRAS, with F-Score of 65.4 and 65.5 % respectively, which is nearly 7.5 % better than no stemming. YASS, LOVIN, and SNS performed almost equal with an increase of nearly 5 % in F-Score as compared to no stemming baseline. The performance of HPS in the classification task is reported to be slightly less than other rule-based or statistical techniques.

7 Stemming applications

Text stemming algorithms developed in various studies have been applied in various information retrieval, extraction, and other natural language processing applications. Researchers have been using stemming as a pre-processing tool in text and document analysis so as to improve the performance of their systems. In this section, we describe some areas that make use of stemming for text analysis in their pre-processing stage.

- *Machine translation systems* Machine translation, an important area of natural language processing and computational linguistics make use of computer technology to convert the text or speech in one language to another. Improved MT output quality is obtained by removing morphological variations through stemming. Lavie et al. (2004) verified that stemming is advantageous not only to simple evaluation metrics like precision and recall but also to widely used MT evaluation metrics like NIST and BLEU. Toutanova et al. (2008) integrated morphological generation model based on stemming and morphological analysis with a statistical machine translation system, thereby improving the performance of both syntax and phrase-based machine translation. There have been various studies (Zollmann et al. 2006; Ramanathan et al. 2008; Bisazza and Federico 2009) which report that in languages with complex morphology, stemming reduces not only the size of the training data for statistical MT systems but also reduces the out-of-vocabulary rates by a significant percentage.
- *Text summarization systems* Text summarization, particularly automatic summarization is one of the challenging tasks in human communication technology. Generating automated summaries that are close to human-generated summaries is the most challenging

and demanding task in the field. Subsequent studies (Méndez-Cruz et al. 2013; Louis and Nenkova 2009) have shown that various pre-processing steps such as stop word removal, stemming of words and normalization of words improve the performance and computational complexity of the summarization systems.

- *Question answering systems* Text Stemming techniques have also produced a positive effect in the retrieval of documents in the context of QA systems which provide appropriate answers to the questions posed by the user. Monz (2003) studied the effect of stemming on QA system and demonstrated that stemming increase both recall and precision if employed at the indexing time. Fareed et al. (2013) used Khoja stemmer during both indexing and retrieval time and achieved significant improvements in the performance of the QA system.
- *Text classification and clustering* Text classification systems aim at automatically assigning pre-defined labels to unknown documents and are widely used in applications like emails filtering, document indexing, web browsing, etc. Stemming simplifies the classification model by reducing the feature sets of the documents when document words are considered as features. Biba and Gjatu (2014) demonstrated the effect of stemming of compound words in boosting the accuracy of text categorization. Stemming is also found to be beneficial for text clustering where the similar documents are clustered together, but the labels are not pre-defined as in the case of text classification. Researchers have been using stemming (Rosell 2003; Froud et al. 2010; Sandhya et al. 2011) to improve the quality of clusters by reducing the keywords of the documents to the stems, thereby reducing the feature set in a bag of words representation of documents.
- *Text searching* In text searching the major issue is that the user search query is not often correctly formulated to provide the best results. But with stemming, the searcher need not care about the exact word forms of the query terms as stemming help in expanding the query terms by considering all the morphological variants of the terms. Rather than performing blind stemming during query expansion, stemmers are now developed (Peng et al. 2007; Paik et al. 2013 that consider the context of query terms to return the variants that are thematically coherent with the original query. Peng et al. (2007) validated that with context based stemming of query terms, query traffic can be significantly reduced in web search engines.
- *Part of speech tagging* Stemming has been extensively used as a pre-processing tool in POS tagging system, which determines the best syntactic grade of the various words in a particular sentence. Shrivastava and Bhattacharyya (2008) demonstrated that it is possible to develop an efficient POS tagger using stemming and without employing any other morphological analyzers or language resources. Stemmer is useful in the design of POS tagging systems as it reduces the variant words to the root word during the learning process which increases the chance of finding the right choice for the word.
- *Other applications* Beside these applications, stemming is also used in wide range of NLP tools such as word sense disambiguation (Shrivastava et al. 2005), spell checkers (Islam et al. 2007), named entity recognition (Konkol and Konopík 2014), keyword extraction, domain dictionaries, etc. In fact, stemming can be used in any area where text analysis and processing is required to tackle the problems of vocabulary mismatch, reduction in index size, feature set or training data, and thereby improving the performance of the system.

8 Unsupervised stemming: open issues and challenges

There has been plenty of research in the field of stemming since the late sixties. The number of ideas relating to stemming has led to a rapid increase of terms and concepts, which are used interchangeably in the literature of stemming. This has increased the difficulty for the practitioners that which technique is most relevant and which technique should be used for a particular application. Currently, stemming techniques belong to either rule-based category or statistical. As already discussed in the paper; rule-based techniques are time demanding and require various language dependent resources. So, for resource constrained languages, rule-based stemmers are either unavailable or lack complete coverage. Unsupervised statistical stemmers provide an appropriate solution to this problem. But unsupervised stemming also has many issues that need to be addressed. Some of the issues related to unsupervised stemming are discussed in this section.

One major issue in the field of unsupervised stemming is that most of the statistical stemmers are built only for suffixing languages. They only focus on the inflectional variations caused by just adding the suffixes whereas a large number of variations in the words are caused by changing significant parts of the words, compounding of words or even by spelling variations. The rules that result in these changes also occur quite frequently in the large corpus of the languages. So some ways need to be discovered to identify these variations from the corpus, and then accordingly rules need be elaborated. This could lead to correction of many current mistakes or errors in stemming and thus improve the performance of automatic stemming.

Another issue is to develop ways to model advanced semantic relations from the corpus. Currently, most of the stemmers use co-occurrence information, context or distribution similarity from the corpus. Other semantic relations such as semantic spaces must be learned from the corpus so as to improve the performance of the stemmer. This would help the stemmer to decide that in a particular situation the affix can be stripped off or not rather than removing the known affix in each case. Consider the pair of words (1) blue and blues (2) eat and eats which are lexically same and just differ in the suffix ‘s’. The removal of ‘s’ in (1) case will be a stemming mistake as they are semantically different words but removing ‘s’ in (2) case will be a correct approach. All such errors can be resolved by identifying correct semantic information from the corpus.

Another important matter in question of unsupervised stemming is that majority of statistical stemmer requires corpus dependent parameters or threshold settings due to which the stemming procedure is not fully automated. The parameters required by stemming algorithms are dependent upon the type of similarity measure used, nature of corpus, nature of language and many other factors. The future research in the field must explore techniques where the parameters are either static or unsupervised parameter learning, and selection criteria are employed.

The evaluation of stemmers independent of information retrieval process is also an important question to be answered. Researchers have provided metrics like recall, precision, etc. that help the user in deciding a stemmer for a particular application. These metrics are highly dependent on the other tasks involved in the information retrieval chain. Stemming just matches the variants to the root words to represent the document words to concepts, but other processes in the IR chain translate these concepts to multidimensional vectors so as to figure out the similarity between the documents and query terms. The comparison of metrics obtained from IR experiments is quite difficult as it cannot be assured that metrics and techniques used in experiments, other than stemming algorithm, are equivalent. In order

to avoid the involvement of other processes in IR chain, some direct evaluation approaches are also proposed in the literature such as reduction in index size or under-stemming and over-stemming errors which ultimately define the stemmer strength. But these evaluations have been done in a relative way, and researchers are unable to provide an absolute and objective metric for stemmer strength.

9 Conclusion

The research in the field of text stemming is continuing in many directions. The current activities in this field have been driven by the challenge of developing unsupervised statistical stemmers that do not involve any language-dependent features. In this article, we have made an attempt to review various text stemming algorithms and present the current state-of-the-art. We began by classifying the various stemming techniques with reference to previous work. Subsequently, a survey of more than hundred linguistic (rule-based) and statistical (unsupervised) stemmers is presented. In the category of linguistic stemmers, 30 different languages are covered belonging to nine different language families. We also critically examined the retrieval performance of sixteen well-known linguistic and statistical stemmers on standard TREC, CLEF, and FIRE collections to identify their strengths, weaknesses, and significance. The performance of stemmers is also analyzed according to a number of factors such as different indexing schemes, different size, and nature of corpus, different tasks, etc. Further, certain open issues and challenges in unsupervised stemming are discussed. The issues like learning advanced semantic relations from the corpus, discovering rules other than affix stripping, unsupervised parameter tuning, evaluation independent of IR system, etc. need to be addressed by the research community so as further to improve the performance of unsupervised language independent stemmers.

References

Adam G, Asimakis K, Bouras C, Poulopoulos V (2010) An efficient mechanism for stemming and tagging: the case of greek language. In: Proceedings of the 14th international conference on knowledge-based and intelligent information and engineering systems, pp 389–397

Ahmad F, Yusoff M, Sembok T (1996) Experiments with a stemming algorithm for Malay words. J Am Soc Inf Sci 47:909–918

Ahmed F, Nürnberger A (2009) Evaluation of n-gram conflation approaches for Arabic text retrieval. J Am Soc Inf Sci Technol 60:1448–1465

Akram Q-A, Naseer A, Hussain S (2009) Assas-band, an affix-exception-list based Urdu stemmer. In: Proceedings of the 7th workshop on Asian language resources, pp 40–46

Aljlayl M, Frieder O (2002) On Arabic search: improving the retrieval effectiveness via a light stemming approach. In: ACM eleventh conference on information and knowledge management, pp 340–347

Al-Kabi M (2013) Towards improving Khoja rule-based Arabic stemmer. In: IEEE Jordan conference on applied electrical engineering and computing technologies (AEECT), pp 1–6

Alshalabi R (2005) Pattern-based stemmer for finding Arabic roots. Inf Technol J 4:38–43

Al-Shalabi R, Kannan G, Hilat I et al (2005) Experiments with the successor variety algorithm using the cutoff and entropy methods. Inf Technol J 4:55–62

Al-shammari E, Lin J (2008) Towards an error-free Arabic stemming. In: Proceedings of the 2nd ACM workshop on improving non English web searching, iNEWS'08, pp 9–16

Alvares R, Garcia A, Ferraz I (2005) STEMBR: a stemming algorithm for the Brazilian Portuguese language. In: Proceedings of 12th Portuguese conference on artificial intelligence, EPIA 2005, pp 693–701

Al-Zyoud A, Al-Rabayah W (2015) Arabic stemming techniques: comparisons and new vision. In: Proceedings of the 8th IEEE GCC conference and exhibition, pp 1–6

Amati G (2006) Frequentist and bayesian approach to information retrieval. In: Advances in information retrieval. Springer, pp 13–24

Amati G, Van Rijsbergen CJ (2002) Probabilistic models of information retrieval based on measuring the divergence from randomness. ACM Trans Inf Syst 20:357–389
Apache lucene. http://lucene.apache.org
Baayen RH, Piepenbrock R, van H R (1993) The CELEX lexical data base (CD-ROM). Linguistic data consortium. University of Pennsylvania, Philadelphia
Bacchin M, Ferro N, Melucci M (2002) The effiectiveness of a graph-based algorithm for stemming. In: Digital libraries: people, knowledge, and technology. Springer, pp 117–128
Bacchin M, Ferro N, Melucci M (2005) A probabilistic model for stemmer generation. Inf Process Manag 41:121–137
Baeza-Yates R, Ribeiro-Neto B (2011) Modern information retrieval: the concepts and technology behind search, 2nd edn. ACM Press, Los Angeles
Baroni M, Matiasek J, Trost H (2002) Unsupervised discovery of morphologically related words based on orthographic and semantic similarity. In: Workshop on morphological and phonological learning (MPL'02), pp 48–57
Bhamidipati NL, Pal SK (2007) Stemming via distribution-based word segregation for classification and retrieval. IEEE Trans Syst Man Cybern B Cybern 37:350–360
Bhattacharya S, Chhoudhury M, Sarkar S, Basu A (2005) Inflectional morphology synthesis for Bengali noun, pronoun and verb systems. In: Proceedings of the national conference on computer processing of Bangla, pp 34–43
Biba M, Gjatu E (2014) Boosting text classification through stemming of composite words. Recent Adv Intell Inform 235:185–194
Bisazza A, Federico M (2009) Morphological pre-processing for Turkish to English statistical machine translation. In: International workshop on spoken language translation, pp 129–135
Braschler M, RippLinger B (2004) How effective is stemming and decompounding for German text retrieval? Inf Retr Boston 7:291–316
Brychcín T, Konopík M (2015) HPS: high precision stemmer. Inf Process Manag 51:68–91
Carlberger J, Dalianis H, Hassel M, Knutsson O (2001) Improving precision in information retrieval for Swedish using stemming. In: Proceedings of 13th Nordic conference on computational linguistics (NODALIDA '01)
Chan E (2006) Learning probabilistic paradigms for morphology in a latent class model. In: Proceedings of the eighth meeting of the ACL special interest group on computational phonology and morphology, pp 69–78
Chaupattnaik S, Nanda S, Mohanty S (2012) A suffix stripping algorithm for Odia stemmer. Int J Comput Linguist Nat Lang Process 1:1–5
Chen A, Gey F (2002) Building an Arabic stemmer for information retrieval. In: Proceedings of the text retrieval conference (TREC'02), pp 631–639
Cilden E (2006) Stemming Turkish words using snowball. http://snowball.tartarus.org/algorithms/turkish/stemmer.html
Darwish K, Oard D (2002) CLIR experiments at Maryland for TREC-2002: Evidence combination for Arabic-English retrieval. In: Proceedings of the text retrieval conference (TREC'02), pp 703–710
Das A, Bandyopadhyay S (2010) Morphological stemming cluster identification for Bangla. In: Knowledge sharing event-I: task 3: morphological analyzers and generators, Mysore
Dasgupta S, Khan M (2004) Feature unification for morphological parsing in Bangla. In: Proceedings of the 7th international conference on computer and information technology
Dawson JL (1974) Suffix removal for word conflation. Bull Assoc Lit Linguist Comput 2:33–46
Deepamala N, Kumar P (2015) Kannada stemmer and its effect on Kannada documents classification. In: Proceedings of the international conference on computational intelligence in data mining, pp 75–86
Dolamic L, Savoy J (2009a) Indexing and stemming approaches for the Czech language. Inf Process Manag 45:714–720
Dolamic L, Savoy J (2009b) Indexing and searching strategies for the Russian language. J Am Soc Inf Sci Technol 60:2540–2547
Dolamic L, Savoy J (2010) Comparative study of indexing and search strategies for the Hindi, Marathi, and Bengali languages. ACM Trans Asian Lang Inf Process 9:11
El-Beltagy S, Rafea A (2011) An accuracy-enhanced light stemmer for Arabic text. ACM Trans Speech Lang Process 7:1–22
Elrajubi O (2013) An improved Arabic light stemmer. In: 3rd International conference on research and innovation in information systems (ICRIIS'13), pp 33–38
Eryiğit G, Adalı E (2004) An affix stripping morphological analyzer for Turkish. In: Proceedings of the IASTED international conference artificial intelligence and applications

Fareed NS, Mousa HM, Elsisi AB (2013) Enhanced semantic Arabic Question answering system based on Khoja stemmer and AWN. In: 9th international computer engineering conference (ICENCO). IEEE, Giza, pp 85–91
Fernández A, Díaz J, Gutiérrez Y (2011) An unsupervised method to improve Spanish stemmer. In: Natural language processing and information systems. Springer, pp 221–224
Figuerola C, Gomez R, Rodriguez A, Berrocal J (2001) Stemming in Spanish: a first approach to its impact on information retrieval. In: Working notes of CLEF 2001 workshop. Darmstadt, Germany, pp 197–202
Frakes WB (1992) Stemming algorithms. In: Frakes WB, Baeza-Yates R (eds) Information retrieval: data structures and algorithms. Prentice-Hall, Upper Saddle River, New Jersey, pp 131–160
Frakes WB, Fox CJ (2003) Strength and similarity of affix removal stemming algorithms. ACM SIGIR Forum 37:26–30
Froud H, Benslimane R, Lachkar A, Ouatik SA (2010) Stemming and similarity measures for Arabic documents clustering. In: 5th International symposium on communications and mobile network (ISVC), pp 1–4
Ganguly D, Leveling J, Jones G (2012) DCU@FIRE-2012: rule-based stemmers for Bengali and Hindi. In: Fourth workshop of the forum for information retrieval evaluation (FIRE 2012)
Gaustad T, Bouma G, Groningen R (2002) Accurate stemming of Dutch for text classification. Lang Comput 45:104–117
Goldsmith J (2001) Unsupervised learning of the morphology of a natural language. J Comput Linguist 27:153–198
Goldsmith J (2006) An algorithm for the unsupervised learning of morphology. Nat Lang Eng 12:353–371
Gupta V (2014) Hindi rule based stemmer for nouns. Int J Adv Res Comput Sci Softw Eng 4:62–65
Gupta V, Lehal GS (2011) Punjabi language stemmer for nouns and proper names. In: Proceedings of the 2nd workshop on south and southeast Asian natural language processing (WSSANLP), pp 35–39
Hafer MA, Weiss SF (1974) Word segmentation by letter successor varieties. Inf Storage Retr 10:371–385
Hammarström H, Borin L (2011) Unsupervised learning of morphology. Comput Linguist 37:309–350
Harman D (1991) How effective is suffixing? J Am Soc Inf Sci 42:7–15
Harmanani H, Keirouz W, Raheel S (2006) A rule-based extensible stemmer for information retrieval with application to Arabic. Int Arab J Inf Technol 3:265–272
Hegde Y, Kadambe S, Naduthota P (2013) Suffix stripping algorithm for Kannada information retrieval. In: International conference on advances in computing, communications and informatics (ICACCI), pp 527–533
Hiemstra D (2001) Using language models for information retrieval. Taaluitgeverij Neslia Paniculata
Honrado A, Leon R, O'Dennol R, Sinclair D (2000) A word stemming algorithm for the Spanish language. In: Proceedings of the 7th international symposium on string processing and information retrieval, pp 139–145
Huddleston R (1988) English grammar: an outline. Cambridge University Press, Cambridge
Hull DA (1996) Stemming algorithms—a case study for detailed evaluation. J Am Soc Inf Sci 47:70–84
Islam M, Uddin M, Khan M (2007) A Light weight stemmer for Bengali and its use in spelling checker. In: Proceedings of the 1st international conference on digital communications and computers
Jivani AG (2011) A comparative study of stemming algorithms. Int J Comput Technol Appl 2:1930–1938
Joachims T (1998) Text categorization with support vector machines: learning with many relevant features. In: Proceedings of 10th European conference on machine learning, chapter. Springer, pp 137–142
Jordan C, Healy J, Keselj V (2006) Swordfish: an unsupervised ngram based approach to morphological analysis. In: Proceedings of the 29th annual international ACM SIGIR conference on research and development in information retrieval, pp 657–658
Jurka TP, Collingwood L, Boydstun AE et al (2013) RTextTools: a supervised learning package for text classification. R J 5:6–12
Kalamboukis T, Nikolaidis S (1995) Suffix stripping with modern Greek. Progr Electron Libr Inf Syst 29:313–321
Kalamboukis T, Nikolaidis S (1999) An evaluation of stemming algorithms with modern Greek. In: Proceedings of the 7th Hellenic conference on informatics, pp 61–70
Kchaou Z, Kanoun S (2008) Arabic stemming with two dictionaries. In: IEEE international conferenece on innovations in information technology, pp 688–691
Khoja S, Garside R (1999) Stemming Arabic text. Computing Department, Lancaster University, Lancaster
Kleinberg J (1999) Authoritative sources in a hyperlinked environment. J ACM 46:604–632
Konkol M, Konopík M (2014) Named entity recognition for highly inflectional languages: effects of various lemmatization and stemming approaches. In: Text, speech and dialogue, pp 267–274
Korenius T, Laurikkala J, Jarvelin 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 (CIKM'04), pp 625–633

Kraaij W, Pohlman R (1994) Porter's stemming algorithm for Dutch. New Rev Doc Text Manag 1:25–43
Kraaij W, Pohlman 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
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, pp 191–202
Kumar D, Rana P (2010) Design and development of a stemmer for Punjabi. Int J Comput Appl 11:18–23
Larkey L, Ballesteros L, Connell ME (2007) Light stemming for Arabic information retrieval. Arab Comput Morphol Text Speech Lang Technol 38:221–243
Larkey L, Ballesteros L, Connell ME (2002) Improving stemming for Arabic information retrieval: light stemming and co-occurrence analysis. In: Proceedings of the 25th annual international ACM conference on research and development in information retrieval (SIGIR '02), pp 275–282
Larkey L, Connell M, Abdulijaleel N (2003) Hindi CLIR in thirty days. ACM Trans Asian Lang Inf Process 2:130–142
Lavie A, Sagae K, Jayaraman S (2004) The significance of recall in automatic metrics for MT evaluation. In: Machine translation: from real users to research. Springer, pp 134–143
Lennon M, Peirce DS, Tarry BD, Willett P (1981) An evaluation of some conflation algorithms for information retrieval. J Inf Sci 3:177–183
Lennon M, Pierce DS, Tarry BD, Willett P (1988) An evaluation of some conflation algorithms for information retrieval. In: Document retrieval systems, pp 99–105
Louis A, Nenkova A (2009) Automatically evaluating content selection in summarization without human models. In: Proceedings of the conference on empirical methods in natural language processing, pp 306–314
Lovins JB (1968) Development of a stemming algorithm. Mech Transl Comput Linguist 11:22–31
Lushanthan S, Weerasingha A, Hearth D (2014) Morphological analyzer and generator for Tamil language. In: International conference on advances in ICT for emerging regions (ICTer), pp 190–196
Mass D (1996) MPROE—Ein system zur analyse und synthese deutscher Wörter. In: Hauser R (Ed) Linguistische Verifikation. Max Niemeyer Verlag, Tübingen
Mahmud M, Afrin M, Razzaque M et al (2014) A rule based Bengali stemmer. In: International conference on advances in computing, communication and informatics, pp 2750–2756
Majumder P, Mitra M, Datta K (2007a) Statistical vs. rule-based stemming for monolingual french retrieval. Eval Multiling Multi Modal Inf Retr 4730:107–110
Majumder P, Mitra M, Parui SK et al (2007b) YASS: yet another suffix stripper. ACM Trans Inf Syst 25:18
Majumder P, Mitra M, Pal D (2008) Bulgarian, Hungarian and Czech stemming using YASS. In: Advances in multilingual and multimodal information retrieval, pp 49–56
Manning CD, Raghavan P, Schutze H (2008) Introduction to information retrieval. Cambridge University Press, New York
Mayfield J, Mcnamee P (2003) Single N-gram stemming. In: Proceedings of the 26th annual international ACM SIGIR conference on research and development of information retrieval, pp 415–416
Mcnamee P, Mayfield J (2004) Character n-gram tokenization for European language text retrieval. Inf Retr Boston 7:73–97
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 (CIKM'03), pp 131–138
Méndez-Cruz C-F, Torres-Moreno J-M, Medina-Urrea A, Sierra G (2013) Extrinsic evaluation on automatic summarization tasks: testing affixality measurements for statistical word stemming. In: Advances in computational intelligence. Springer, pp 46–57
Meyer D, Dimitriadou E, Hornik K et al (2012) Misc functions of the department of statistics (e1071), TU Wien. R Packag 1:5–24
Monz C (2003) From document retrieval to question answering. Institute for Logic, Language and Computation, Amsterdam
Monz C, Rijke M (2002) Shallow morphological analysis in monolingual information retrieval for Dutch, German, and Italian. Eval Cross Lang Inf Retr Syst 2046:262–277
Moral C, Antonio A, Imbert R, Ramirez J (2014) A survey of stemming algorithms in information retrieval. Inf Res 19:1–14
Nakov P (2003) Design and evaluation of inflectional stemmer for Bulgarian. In: Proceedings of workshop on Balkan language resources and tools
Ntais G (2006) Development of a stemmer for the Greek language. Master Thesis, Department of Computer and Systems Sciences, Stockholm University

Oard D, Levow G, Cabezas C (2001) CLEF experiments at Maryland? Statistical stemming and backoff translation. In: Proceedings of the workshop of cross-language evaluation forum on cross language information retrieval and evaluation. Springer, Berlin, pp 176–187

Open American National Corpus. http://www.anc.org/data/oanc

Orengo V, Huyck C (2001) A stemming algorithm for the Portuguese language. In: Proceedings of 8th internatioanl symposium on string processing and information retrieval, pp 186–193

Othman R (1993) Footer Malay word for document retrieval system. M.Sc. Thesis. National University of Malaysia

Ounis I, Amati G, Plachouras V, et al (2006) Terrier: a high performance and scalable information retrieval platform. In: Proceedings of ACM SIGIR'06 workshop on open source information retrieval (OSIR 2006)

Paice CD (1990) Another stemmer. ACM SIGIR Forum 24:56–61

Paice CD (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

Paik J, Mitra M, Parui S, Jarvelin K (2011a) GRAS: an effective and efficient stemming algorithm for information retrieval. ACM Trans Inf Syst 29:1–24

Paik JH, Pal D, Parui SK (2011c) A novel corpus-based stemming algorithm using co-occurrence statistics. In: Proceedings of the 34th annual international ACM SIGIR conference on research and development in information retrieval (SIGIR'11). ACM, New York, pp 863–872

Paik JH, Parui SK (2011b) A fast corpus-based stemmer. ACM Trans Asian Lang Inf Process 10:1–16. doi:10.1145/1967293.1967295

Paik JH, Parui SK, Pal D, Robertson SE (2013) Effective and robust query-based stemming. ACM Trans Inf Syst 31:1–29. doi:10.1145/2536736.2536738

Patel P, Popat K, Bhattacharyya P (2010) Hybrid stemmer for Gujarati. In: Proceedings of the 23rd international conference on computational linguistics (COLING), pp 51–55

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—SIGIR '07. ACM Press, New York, p 639

Popovic M, Willet P (1992) The effectiveness of stemming for natural-language access to Slovene textual data. J Am Soc Inf Sci 43:384–390

Porter MF (1980) An algorithm for suffix stripping. Progr Electron Libr Inf Syst 14:130–137

Porter MF (2001) Snowball: a language for stemming algorithms. http://snowball.tartarus.org

Ramachandran V, Krishnamurthi I (2012) An iterative stemmer for Tamil language. In: Proceedings of the 4th Asian conference, ACIIDS 2012, pp 197–205

Ramanathan A, Hegde J, Shah RM, et al (2008) Simple syntactic and morphological processing can help English–Hindi statistical machine translation. In: International joint conference on natural language processing, pp 513–520

Ramanathan A, Rao D (2003) A lightweight stemmer for Hindi. In: Proceedings of the 10th conference of the European chapter of the association for computational linguistics

Robertson SE, Walker S, Beaulieu M (2000) Experimentation as a way of life: Okapi at TREC. Inf Process Manag 36:95–108

Rosell M (2003) Improving clustering of Swedish newspaper articles using stemming and compound splitting. In: NoDaLiDa 2003, Reykjavik, Iceland 2003, pp 1–7

Salton G, McGill M (1971) The SMART retrieval system—experiments in automatic document retrieval. Prentice Hall Inc., Englewood Cliffs

Sandhya N, Lalitha YS, Sowmya V et al (2011) Analysis of stemming algorithm for text clustering. IJCSI Int J Comput Sci Issues 8:352–359

Savoy J (1999) A stemming procedure and stopword list for general French corpora. J Am Soc Inf Sci 50:944–952

Savoy J (2006) Light stemming approaches for the French, Portuguese, German and Hungarian languages. In: Proceedings of the 2006 ACM symposium on applied computing, pp 1031–1035

Savoy J (2008) Searching strategies for the Hungarian language. Inf Process Manag 44:310–324

Savoy J, Berger P-Y (2006) Monolingual, Bilingual, and GIRT information retrieval at CLEF-2005. In: 6th workshop of the cross-language evalution forum, CLEF 2005, pp 131–140

Sembok T (2005) Word stemming algorithms and retrieval effectiveness in Malay and Arabic documents retrieval systems. In: Proceedings of the world academy of science, engineering and technology

Sever H, Bitirim Y (2003) FindStem: analysis and evaluation of a Turkish stemming algorithm. In: Proceedings of the 10th international symposium on string processing and information retrieval, pp 238–251

Sharifloo A, Shamsfard M (2008) A bottom up approach to persian stemming. In: Proceedings of the third international joint conference on natural language processing

Shrivastava M, Bhattacharyya P (2008) Hindi POS tagger using naive stemming: harnessing morphological information without extensive linguistic knowledge. In: Proceedings of international conference on NLP (ICON08)

Shrivastava M, Mohapatra B, Bhattacharyya P et al (2005) Morphology based natural language processing tools for Indian languages. In: Proceedings of the 4th annual international research student seminar in computer science

Smirnov I (2008) Overview of stemming algorithms. In: Mechanical Translation. http://thesmirnovs.org/info/stemming.pdf. Accessed 25 May 2014

Soares MVB, Prati RC, Monard MC (2009) Improvement on the Porter's stemming algorithm for Portuguese. IEEE Lat Am Trans 7:472–477

Stein B, Potthast M (2007) Putting successor variety stemming to work. In: Advances in data analysis. Springer, pp 367–374

Suba K, Jiandani D, Bhattacharyya P (2011) Hybrid inflectional stemmer and rule-based derivational stemmer for Gujarati. In: Sangal R, Malik M (eds) Proceedings of the 23rd workshop on south and southeast Asian natural language processing (WSSANLP). Asian Federation of Natural Language Processing, Chiang Mai, Thailand, pp 1–8

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), pp 152–157

Tai S, Ong C, Abdullah N (2000) On designing an automated Malaysian stemmer for the Malay language. In: Proceedings of the fifth international workshop on information retrieval with Asian languages, pp 207–208

Tala F (2003) A study of stemming effects on information retrieval in Bahasa Indonesia. Master Thesis, University of Amsterdam

Terrier information retrieval platform. http://terrier.org

The lemur project. http://www.lemurproject.org

The R project for statistical computing. http://www.r-project.org

Toutanova K, Suzuki H, Ruopp A (2008) Applying morphology generation models to machine translation. In: Association for computational linguistics, pp 514–522

Xapian project website. http://xapian.org

Xu J, Croft WB (1998) Corpus-based stemming using cooccurrence of word variants. ACM Trans Inf Syst 16:61–81

Yadav A, Yadav R, Pal S (2012) ISM@FIRE-2012 adhoc retrieval and morpheme extraction task. In: Post proceedings of FIRE-2012

Zollmann A, Venugopal A, Vogel S (2006) Bridging the inflection morphology gap for Arabic statistical machine translation. In: Proceedings of the human language technology, pp 201–204