Text Stemming: Approaches, Applications, and Challenges

JASMEET SINGH and VISHAL GUPTA, Panjab University, Chandigarh

Stemming is a process in which the variant word forms are mapped to their base form. It is among the basic text pre-processing approaches used in Language Modeling, Natural Language Processing, and Information Retrieval applications. In this article, we present a comprehensive survey of text stemming techniques, evaluation mechanisms, and application domains. The main objective of this survey is to distill the main insights and present a detailed assessment of the current state of the art. The performance of some well-known rule-based and statistical stemming algorithms in different scenarios has been analyzed. In the end, we highlighted some open issues and challenges related to unsupervised statistical text stemming. This research work will help the researchers to select the most suitable text stemming technique in a specific application and will also serve as a guide to identify the areas that need attention from the research community.

CCS Concepts: • **Information systems** → **Information retrieval**; • **Computing methodologies** → **Natural language processing**; • **General and reference** → *Surveys and overviews*

Additional Key Words and Phrases: Text stemming, morphological analysis, stemmers, rule-based stemming, statistical stemming, corpus-based stemming

ACM Reference Format:
Jasmeet Singh and Vishal Gupta. 2016. Text stemming: Approaches, applications, and challenges. ACM Comput. Surv. 49, 3, Article 45 (September 2016), 46 pages.
DOI: http://dx.doi.org/10.1145/2975608

1. INTRODUCTION

With the huge amount of digital data available in multiple languages, it has become important to develop various language processing tools that could efficiently manage the large document bases. In many Natural Language Processing (NLP) and Information Retrieval (IR) applications, construction of vocabulary of words and language models is an important task. But a large number of morphological variations in the words, especially in morphologically rich languages, pose a great challenge. Text stemming is a useful pre-processing technique that can handle these variations.

Stemming is a linguistic process in which the various morphological variants of the words are mapped to their base forms. For instance, the words *plays, played, playing*, *player*, and *players* are mapped to their base form *play* with the help of stemming. The algorithms or the programs that perform stemming are called stemmers. Stemming is a simple language processing that is found to be quite effective in a number of applications.

Authors' addresses: J. Singh and V. Gupta, Computer Science and Engineering Department, University Institute of Engineering and Technology, Panjab University, Sector 25, Chandigarh, India-160014; emails: jasmeetsingh.gagneja@gmail.com, vishal@pu.ac.in.

DOI: http://dx.doi.org/10.1145/2975608

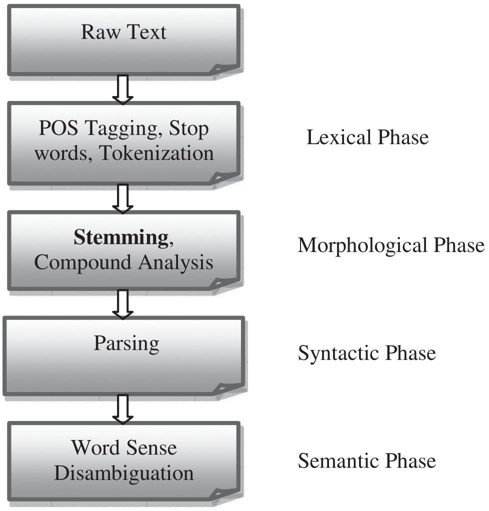
Raw Text
POS Tagging, Stop words, Tokenization
Lexical Phase
Stemming, Compound Analysis
Morphological Phase
Parsing
Syntactic Phase
Word Sense Disambiguation
Semantic Phase

Fig. 1. Role of stemming in the text processing chain for Information Retrieval.

1.1. Stemming Algorithms: Purpose

Stemming algorithms are designed for a specific application, and they serve different purposes in different types of tasks. In Information Retrieval (IR), the purpose o stemming is twofold. First, it matches the variant word forms in the documents and the queries to the base word form, thereby solving the problem of vocabulary mismatch. It helps in improving the performance of Information Retrieval systems in terms of both precision and recall. Thus, stemming is used as a part of the user interface in retrieval systems where users can write any variant of the word in the query. Second, the conflation of the words having the same stem into single class reduces the size of the dictionary. It helps in reducing the storage space required to store the various structures (such as index of terms) of an IR system. In Information Retrieval, stemming is applied at an early phase in the text pre-processing chain, as shown in Figure 1.

Stemming methods have also been incorporated in the statistical language modeling framework. In statistical language modeling, stemming increases the frequency of occurrence of the words by mapping the variant forms to the same stem. So stemming can be regarded as a kind of smoothing of probability estimates. Various studies [Allan and Kumaran 2003; Brychcín and Konopík 2015] have shown that stemming helps in improving the quality of the language model. The improvement in language model leads to improvement in the task where the model is being used. Stemming can also be viewed as a method of query expansion. The various terms of the user query can be enhanced with the word forms not used by the user. From another perspective, stemming can also be viewed as a mechanism for word sense disambiguation since the different concepts used in the query are normalized to a single principle concept according to the context in the query [Krovetz 1993].

1.2. Basic Concepts

—*Stem*. Stem is the basic word form that is modified to obtain variant word forms using different linguistic processes such as affixation (addition of affixes), compounding (combination of base forms), and so on [Huddleston 1988; Sampson 2005]. For example, the stem of the word *friendships* is *friend,* to which *–ship* (derivational suffix) is first attached to form a new variant *friendship,* and then *–s* (inflectional suffix) is attached.

—*Lemma and Lexeme.* Lexemes correspond to a collection of all the word forms that have the same meaning and lemma is one specific form that is selected to represent the lexeme. Lemma is the word form that matches the dictionary. For example, the words *go, going, gone, goes,* and *went* are forms of the same lexeme represented by the lemma *go.*

—*Inflectional and Derivational Morphology.* Inflectional morphology involves changes in the word caused due to the syntax. The inflectional affixes are added to the words to guarantee that the form of the word is appropriate and the sentence is grammatically correct. They do not change the meaning or part of speech of the word. In English, the inflectional variations include variations in verbs (*recommend* to *recommended*), nouns (possessive forms *'s* in *John's*, plural forms *–s* in *sings*), and adjectives (such as comparative –r and superlative –er in *larger, largest*). Derivational morphology, on the other hand, involves variations in word forms that change the meaning or function of the word. These variations make new and different lexemes (for example, *sad* to *sadly*). They may or may not change the part of speech of the word. Inflectional suffixes are added at last after any derivational ending and once they have been added no more derivational affixes can be added.

—*Light and Aggressive Stemming.* Light and aggressive stemming are the terms used in the context of the software that does not make use of a lexicon. It is a measure of how to compare different programs with regard to the degree of truncation involved. Light stemming methods conflate simple inflected forms. They can handle the changes in number, person, gender, tense, case, mood, and so on. Aggressive stemming methods can also handle derivational variations in the word forms. They can detect changes in the part of speech and can map the word forms to the forms from which they are derived.

—*Stemming and Lemmatizing.* Stemming and Lemmatizing are both used in NLP applications to reduce the variant word forms to their base forms. The basic difference between stemming and lemmatization is that stemming can include derivational variants, and it is usually discussed in the context of the application (usually IR), whereas lemmatization is usually discussed *independently* of the application.

1.3. Organization of the Article

The rest of the article is organized as follows. In Section 2, the various stemming methods are categorized according to certain key parameters. In Section 3, evaluation mechanisms of stemming methods are presented. Various rule-based stemmers developed for English, European, and Asian languages are described in Section 4. In Section 5, statistical stemming methods are discussed. In Section 6, the performance of some rule-based and statistical stemming methods is compared in Information Retrieval experiments using standard test collections. The comparison of stemmers in Language Modeling and Natural Language Processing applications is presented in Section 7. The applications of stemming in various domains are described in Section 8. In Section 9, the review of various morpho-challenge competitions is presented. In Section 10, some of the challenges related to unsupervised stemming are described. Section 11 concludes this article.

2. CATEGORIZATION OF STEMMING TECHNIQUES

The stemming process has a rich literature, and a number of stemmers of varying flavors have been developed over the decades. Stemming methods may range from simple approaches like the removal of plurals and present and past participles to complex approaches that remove a variety of suffixes and include a lexicon. Current

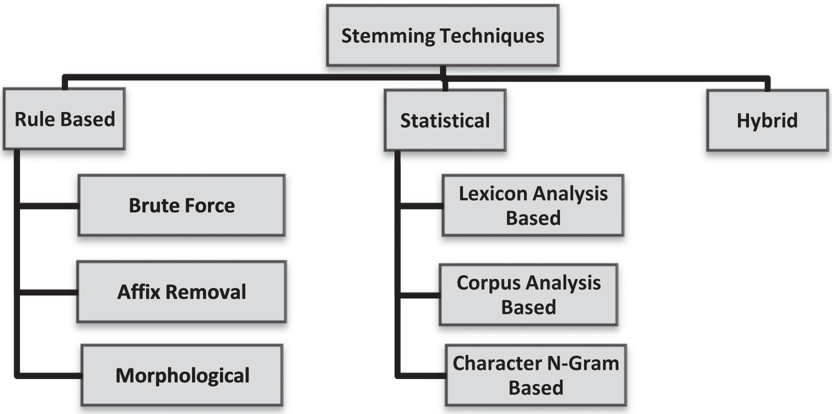

Fig. 2. Categorization of stemming techniques.

stemming algorithms belong to one of three categories: *Rule Based, Statistical,* or *Hybrid*. Each of these categories finds the stems of the variant words in their own typical way. The categorization of these stemming techniques is presented in Figure 2.

2.1. Rule-Based Stemmers

Rule-based stemmers transform the variant word forms into their stems or base forms by using certain pre-defined language-specific rules. So these are also called language-specific stemmers. The creation of language-specific rules requires expertise in language or at least a native speaker of the particular language. Moreover, rule-based stemmers sometimes employ additional linguistic resources like dictionaries to conflate morphologically related words. The major advantage of rule-based stemmers is their ease of use, that is, the language-specific rules once created can be applied to any corpus without any additional processing. But for languages where the resources are poor, these stemmers are not preferred. These stemmers tend to be better than statistical stemmers in applying complex morphological rules of the language [Brychcín and Konopík 2015]. Rule-based stemmers are further classified into following three categories:

(1) *Brute-Force Algorithms.* Brute-force stemmers make use of a lookup table to return the stem of the word. This lookup table maintains relations between the variant words and their root forms. The table is checked to find the matching inflection, and the associated stem is returned. These stemming techniques are also called table lookup or dictionary-based techniques. One notable advantage of these stemmers is that they can handle the inflected word forms of a language that do not obey the language-specific rules appropriately. For instance, suffix removal algorithms can stem the word "*eating*" to "*eat*" but it cannot stem the alternate inflection "*ate.*" The major limitation of these algorithms is that all the variant words cannot be manually collected and recorded in a lookup table. So it cannot stem the words that are not present in the table. Moreover, it consumes a lot of space to store the list of relations.

(2) *Affix Removal Algorithms.* Affix refers to suffix or prefix of a word. So, as the name suggests, these techniques remove the suffix and/or prefix from the variant word forms. The stemmers in this category make use of a suffix/prefix list along with certain context-sensitive rules to obtain the stem. Most of the work is done on suffix removal as compared to prefixes. One major weakness of these stemmers is

Table I. Comparison of Rule-Based and Statistical Stemmers

Feature	Rule-Based Stemmers	Statistical Stemmers
Use of Linguistic Resources (suffix list, dictionary, etc.)	✓	×
Use of Corpus	×	✓
Need of Linguistic Experts	✓	×
Involve Computations	×	✓

that the stems produced after removal of suffixes are not real words of the language. These truncated word forms are poor for human interfaces and present difficulties in certain applications. The problem depends on how the transformation is being used. For example, if we use the stems to create clusters of words, then the failure to identify a word is not necessarily harmful but in applications like word-sense disambiguation, these stems cannot be used, as it is not possible to resolve the meaning of the word without knowing the word we are dealing with. Moreover, affix removal algorithms sometimes produce aggressive conflations. For example, the words *"general," "generic," "generous"* are stemmed to the same root "*gener*" by the suffix stripping process.

(3) *Morphological Stemmers.* These stemmers involve inflectional and derivational morphological analysis to perform stemming. They require large language-specific lexicons containing word groups organized by syntactic and semantic variations [Krovetz 1993]. Inflectional analysis can detect changes in word forms due to gender, tense, mood, case, number, person, or voice. Derivational analysis can detect changes in part of speech (POS) and can reduce surface forms to the forms from which they are derived. For instance, "*advancement*" is stemmed to "*advance*" but "*department*" is not stemmed to *"depart"* as both forms have different semantics. The advantages of morphological stemmers are that they produce morphologically correct roots and can handle various exceptional cases. These stemmers handle roots that are out-of-vocabulary by making use of rules as well as a lexicon. The algorithm first finds the root in the lexicon, but if the root is not found and the suffix is productive enough, the word is transformed.

2.2. Statistical Stemmers

Statistical stemmers use unsupervised or semi-supervised training to learn stemming rules from a corpus of a given language. They group morphologically related words using the ambient corpus, thereby obviating the need for language experts or any additional linguistic resource. So these stemmers are also called *language independent* or *corpus-based* stemmers. The major advantage of corpus-based stemmers is that these stemmers can be applied to a new language with very little effort provided the language satisfies the basic assumptions of the stemmer (like variant words should be formed by adding affixes only). Moreover, statistical stemmers can find fewer frequent cases while processing a large corpus of the language. A number of studies [Brychcin and Konopik 2015; Paik et al. 2011a, 2011b, 2013; Majumder et al. 2007] have shown that statistical stemmers are good substitutes to language-specific stemmers, especially for languages where linguistic resources are incomplete. A comparison between features of statistical and rule-based stemmers is presented in Table I. They follow three types of approaches to statistical stemming:

(1) *Lexicon Analysis-Based Stemmers.* These stemmers analyze a set of words obtained from the corpus to group the lexicographically related words. They find probable stems and suffixes using various methods like computing string distances

[Majumder et al. 2007; Fernandez et al. 2011], the frequency of substrings [Oard 2001; Paik and Parui 2011], and so on.

(2) *Corpus Analysis-Based Stemmers*. These stemmers group morphologically related words by analyzing their co-occurrence or context in the corpus. These are based on the fact that words that co-occur in the corpus are a better representative to be merged than words that do not co-occur. As compared to lexicon analysis-based stemmers, they require relatively large corpus to obtain more reliable co-occurrence information [Paik et al. 2013].

(3) *Character N-Gram-Based Stemmers*. These stemmers learn the stemming rules through frequency or probability of n-grams obtained from the words of the corpus. They can handle morphological variations in alphabetic languages. As compared to other methods in this category, n-gram-based stemmers can handle not only inflectional and derivational morphology but also compounding of words or spelling exceptions [Paik et al. 2011b].

2.3. Hybrid Stemmers

Hybrid stemmers combine several approaches to perform stemming. The combination of approaches generally helps in increasing the efficiency of the stemmer. Hybrid stemmers can be formed by different combination methods such as combining various rule-based approaches or combining a rule-based approach with statistical methods. For instance, the efficiency of a suffix stripping algorithm can be increased with table lookups for unusual verb forms (like run/ran) or singular/plural forms. Similarly, the classes generated by the rule-based stemmers can further be refined by using co-occurrence or other corpus-specific information. It helps in solving the problem of aggressive conflation in rule-based stemmers. A variety of hybrid stemmers [Weiss 2005; Shrivastava et al. 2005; Adam et al. 2010; Patel et al. 2010; Mishra and Prakash 2012; Deepamala and Kumar 2014] for different languages have been developed. These stemmers are described in Section 4.

3. STEMMING EVALUATION MECHANISMS

The evaluation of stemmers has always been a debated affair. Different research groups have proposed a number of evaluation metrics to measure the effectiveness or error rates of the stemmer that can be broadly classified as direct or indirect evaluation methods. The various direct and indirect methods of evaluation proposed in the literature are described below.

3.1. Direct Evaluation Methods

Direct evaluation methods measure the performance of the stemmer directly on a collection of testing words independent of any application. These methods measure the stemmer performance in terms of error rates, words correctly stemmed, statistical methods, and so on. These methods require the collection of test words of the language, which involves a lot of manual work.

(1) *Under-Stemming Errors*. Under-stemming, as the name suggests, is the case when the stemmer strips the suffix below the expected level. In these errors, the words that have the same stem are not conflated together. For example, “dentistry” and “dentist” are not stemmed to the same root by the Porter algorithm [Porter 1980]. A high number of under-stemming errors decreases the performance of the stemmer.

Procedures like partial matching used in stemming algorithms [Lovins 1968; Dawson 1974] help in decreasing these errors by conflating the two stems if they are morphologically similar above some defined cutoff. These procedures, in some cases, produce more errors but still they are useful and give good results [Moral et al. 2014]. Paice [1994] proposed that under-stemming errors can be estimated using the Under-Stemming Index (UI), defined as UI = 1-Conflation Index, where conflation index is the ratio of the number of word pairs which are correctly grouped together to the total number of words.

(2) *Over-Stemming Errors.* Over-stemming is the case where the stemmer removes more terminations, thereby truncating parts that belong to the stem of the word. In these errors, two words having different morphological roots are conflated together; for example, "illegal" and "illegible" are both stemmed to "illeg" and are grouped together. Over-stemming errors also decrease the performance of stemmers as two words that are not related but have the same stem are wrongly detected.

These errors can be reduced by imposing constraints like minimum stem length of the resultant stem. Paice suggested a metric, named the Over-Stemming Index (OI), for measuring these errors, which is defined as OI = 1-Distinctness index; where the Distinctness index is the ratio of word pairs not conflated together to the total number of word pairs.

(3) *Statistical Testing:* Hull [1996] suggested that various statistical techniques like ANOVA should be used to check that the difference in performance is significant or not. These methods are useful in stemming procedures as the test queries are a small sample of the entire query sets.

(4) *Accuracy.* The accuracy of the stemmer is measured by comparing the output of the stemmer with the required output and computing the number of words correctly stemmed by the stemmer. Frakes and Fox [2003] proposed the following methods to measure stemmer strength.

(a) *Index Compression Factor.* Stemmers stem a number of variant word forms to a single stem, thereby reducing the size of the index. So, Index Compression Factor measures the decrease in corpus size through stemming. The high value of this parameter indicates strong stemmer strength.

(b) *Mean Number of Words per Conflation Class.* It refers to the mean number of words grouped together in a class, that is, the mean number of words stemmed to the same root word. The high value of this parameter is desirable for better stemmer performance.

(c) *Difference between number of words and stem.* This refers to the number of words being modified by the stemmer. Strong stemmers modify the words more often than weak stemmers.

(d) *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 differ. The Modified Hamming distance between the word and their stem is calculated by adding the Hamming distance to the difference in the length of the word and its stem.

3.2. Indirect Evaluation Methods

Indirect methods measure the performance of the stemmers by using them as a preprocessor of a specific application like Information Retrieval, Text Classification, and so on. The major advantages of these methods are that they do not require tedious manual labor as they make use of various automated tools to measure the performance. But they require various resources such as document collections, query sets, and so on, and are quite sensitive to the type of collection and queries used during testing. In Information Retrieval, tasks following measures indirectly measure the stemmer performance:

(1) *Precision.* The number of relevant documents retrieved is to the total documents retrieved is termed as Precision.

(2) *Recall*. The total number of documents retrieved that are relevant to the total number of relevant documents with respect to the query is termed Recall.
(3) *F*-Score. The weighted mean of precision and recall is termed the F-Score. It is widely used for testing the retrieval accuracy, as it considers both Recall and Precision.
(4) *Average Precision (AvP)*. The mean of Precision and Recall values at various ranks in a ranked list of documents is termed Average Precision.
(5) *Mean Average Precision (MAP)*. This is the mean of Average Precision values across queries. It is quite frequently used in evaluating the retrieval accuracy of an IR system.

Stemming improves precision as well as recall. This is because of the impact on Term Frequency-Inverse Document Frequency (TF-IDF) weighting, in which we get a different frequency by grouping variant word forms. Thus, more relevant documents are promoted at superior ranks.

4. RULE-BASED STEMMERS

The rule-based stemming algorithms are generally developed for a specific application (usually IR). A variety of rule-based stemmers for different languages has been proposed in the literature that makes use of different language-specific rules and linguistic resources such as suffix list, stem tables, dictionaries, tagged data, and so on. Initial work in this field has been done for the development of English stemmers, but with an increase in multi-lingual data in digital form, rule-based stemmers for other languages have also been developed. In the following subsections, we describe rule-based stemmers proposed in the literature grouped according to different languages.

4.1. English Rule-Based Stemmers

English has relatively few inflectional variants compared with other languages. Morphological variants of English language are formed by the addition of prefixes and/or suffixes, combination of two or more words, and formation of new words from existing words. Harman [1991] tested a few stemming algorithms on an English dataset and reported a small effect on the overall performance. Successive studies [Krovetz 1993; Hull 1996; Xu and Croft 1998] reported a positive effect of stemming on ad hoc retrieval tasks. But the improvements in retrieval results are fewer as compared to highly inflective languages like Hungarian, Marathi, and so on. Various rule-based stemmers for English are described below.

4.1.1. Trivial Methods. *Truncate (n)* is a simplest stemming algorithm in which the word is stemmed to the first n characters. All the characters after first n characters are truncated. These are also called n-gram stemmers. The words of length less than n characters remain unaltered. If the value of n is small (one or two), then there are a large number of over stemming errors. So, it is not used in real systems and is of academic interest only but is used as a good baseline for evaluating other stemming algorithms [Paice 1994].

S-Removal stemmer is another simple and trivial stemmer proposed by Harman [1991]. It groups singular and plural words of English nouns. It also reduces some tensed verbs to a root form. For example, "*creates*" is mapped to "*create,*" but it omits "*created*" *and* "*creating.*" Three rules are used to delete plural suffixes to obtain the words in singular form. These rules are applied only when the length of words is three or more characters. Moreover, these rules are applied in an order-dependent way, that is, the first applicable rule is only applied. This stemmer is also not used in real systems, as it can only handle plural inflections.

4.1.2. Lovins Stemmer. The Lovins stemmer was the first published work in the field of stemming by Lovin [1968]. It is a single-pass stemmer that works in two steps. First, it removes the suffixes by performing a lookup on a list of 294 suffixes each associated with 1 of 29 context-sensitive conditions. The suffixes in the list are arranged according to their lengths. In order to stem the word, the suffix list is queried on the basis of the longest-match principle. If the suffix with a satisfying condition is found, then it is removed from the word. For instance, in order to stem the word "rationally," the first suffix that matches in the list is "ationaly" with condition "minimum stem length of 3." This suffix is discarded as the stem will be of length less than 3. The next suffix in the list "ionally" with no constraint on stem length is selected, and the root "rat" is returned. In the next step, the stem is recoded by using another list containing 35 transformation rules to convert the roots into valid English words. Finally, the words whose roots are quite close but not essentially same are grouped together using the partial matching method. The Lovins stemmer is simple and fast, but it missed many common suffixes.

4.1.3. Dawson Stemmer. Dawson [1974] improved the Lovins stemmer by proposing two modifications. First, he extended the suffix list to 1,200 by adding more common suffixes to handle cases that are not present in the Lovins stemmer or usually handled in the recoding step of the Lovins stemmer. Second, he eliminated the recoding phase, as it was not found to be reliable in the Lovins stemmer and often failed to form valid words from the root words. Only a partial matching procedure is now used to conflate words whose stem matches in some limit. The partial matching procedure is not a component of the stemming algorithm and is entirely the responsibility of IR system. Like the Lovins stemmer, the Dawson stemmer is also a fast, simple, and single-pass stemmer, but it is not as popular because there is no record of a complete suffix list in the literature, and it lacks standard implementation.

4.1.4. Porter Stemmer. The Porter stemmer [1980] by now is the most popular and widely used English rule-based stemmer. Porter defined English words as a sequence of vowels and consonants, that is, $[C] [VC]^{m} [V]$, where V and C represents one or more vowels and consonants, respectively, and m is the measure of the word. The Porter algorithm defines 60 rules that are applied to the word to be stemmed in five steps. Each rule of the algorithm is of the form (condition) (suffix) → (resultant suffix). The rule specifies the essential condition in which it is to be applied and how the word is altered to obtain the stem. As an example, rule $(m>0)$ NESS $\rightarrow \phi$ denotes that if a word has ending NESS and the measure of the resultant stem is greater than zero, then we remove the ending. So, according to this rule, the word "goodness" is stemmed to "good." The Porter stemmer is efficient with regard to readability and complexity, but errors like over-stemming (probe/probable) are well known. An improvement to the Porter algorithm, called the Porter2 algorithm, has also been developed (which is available at http://snowball.tartarus.org/algorithms/english/stemmer.html).

The author developed a framework for developing stemmers called Snowball [Porter 2001]. The main objective of this framework is to help the programmers to develop their stemmers in other languages by just defining the rules of the language. The framework is quite useful, and a number of stemmers belonging to different language families (Romanic, Scandinavian, Uralic, and German, etc.) have been developed using this framework.

4.1.5. Paice/Husk Stemmer. Paice/Husk stemmer was proposed by Paice [1990] at Lancaster University and implemented with the help of Gareth Husk. It is an iterative stemmer that applies the same set of rules and suffixes in every loop. The rules are stored in a separate file containing a number of sections; each section corresponds to the last letter of the suffix, which helps in making fast access to the rule table. Each rule of Paice/Husk stemmer contains following five parts, two of which are optional.

—Ending written in reverse order
—Flag ∗ denotes that a word can be stemmed only once, that is, intact flag (optional part)
—A number denoting number of characters that are to be deleted from the word to be stemmed (can be zero).
—A string that can be affixed in place of truncated character (optional part)
—Symbol ">" indicating continuation, that is, a word can further be stemmed in the next loop or symbol "." indicating termination, that is, the final stem of the word is obtained.

For example, the rule "sei3y>" denotes that if the word has the ending "–ies," then we replace these three characters with "y" and then further apply the stemming rule in the next loop to obtain the stem. Like the Porter stemmer, the Paice/Husk stemmer also does not require separate recoding phase, as the rule itself can delete or replace a suffix.

4.1.6. Krovetz Stemmer (KSTEM). Krovetz [1993] pointed out that the various suffix stripping-based methods [Lovins 1968; Porter 1980] ignore the meaning of the word and do not make use of a lexicon. This could lead to many bad conflations such as *police/policy* or *paste/past*. Moreover, the stems produced by these stemmers are not real words (*iteration* is stemmed to *iter).* These stems are not useful in NLP applications like word sense disambiguation, as disambiguation is not possible if we do not know the word we are dealing with. Keeping these limitations in mind, the author proposed an inflectional and derivational stemmer whose stems are words rather than truncated forms, and the different word forms are conflated if their meanings are related.

The inflectional stemmer deals with plural word forms, past-tense forms, and the suffix –ing. The plurals include three suffixes, –ies, –es and –s. In the case of –ies, the letter –s is first removed, and the resultant stem is scanned in the dictionary. If it is found, then the stem is accepted; otherwise, the suffix –ies is replaced with –y. In the case of –es, the character –s is first removed, and the resultant word is checked in the dictionary. If the stem is found, then it is accepted; otherwise, the suffix –es is removed and the dictionary is scanned again. If not found, then it is assumed that the stem ends in –e. In the case of –s, the ending is removed provided the ending is not –ous or double "s." The past-tense forms and suffix –ing are also handled in a similar way. But in these cases, the exception for the words with ending –e need not be considered, and the doubled letter needs to be converted into a single letter if not found in a dictionary (as in the case of *stopped* or *stopping*).

The derivational stemmer groups words with the same meaning. It makes use of the list of the most frequently occurring suffixes and dictionary lookup mechanisms to obtain the stem of the word. The retrieval performance of KSTEM and the Porter stemmer is similar but the combination of the inflectional and derivational approach of KSTEM helps in increasing the number of stems corresponding to the real words as compared to the Porter stemmer.

4.1.7. XEROX Stemmer. XEROX is an inflectional and derivational stemmer proposed by Hull [1996]. It is based on the Xerox linguist containing a lexicon of approximately 77,000 basic word forms, of which nearly 500,000 variant forms are generated. The inflectional stemming process can handle changes in word forms as a result of number (echo → echoes), gender (she → her), tense (walk →walked), and so on. The derivational process converts the variant word forms into their original base forms that are associated in both form and meaning. It removes suffixes, prefixes, and some irregular word forms using the lexicon.

The XEROX stemmer and the KSTEM both handle inflectional and derivational suffixes and make use of a dictionary to return valid words. The XEROX stemmer groups prefixed forms, whereas KSTEM does not, as they change the meaning of the word more than suffixes. The XEROX stemmer also handles pronouns, but those are typically removed by a stopwords list in IR, which is counter to that application.

4.2. Rule-Based Stemmers for European Languages

European languages are morphologically more complex than English, and hence stemming is found to be more advantageous to these languages. A number of studies [Popovic and Willet 1992; Kraaij and Pohlman 1994; Savoy 2006] have revealed that stemming produce significant improvements in IR tasks for European languages. In this subsection, we describe different stemming methods proposed in the literature for 17 European languages belonging to six language families, namely Italic, Slavic, Uralic, Hellenic, Albanian, and Germanic.

4.2.1. Stemming in Italic Languages. The Italic languages have more complex morphology than English with a large number of morphological variations. The major language in this family includes Latin and Romance languages such as French, Portuguese, Italian, Spanish, and Romanian. The major stemming work in different languages belonging to the Italic family is described below.

—*French Stemming Methods*. In the French language, the first rule-based stemming method was proposed by Savoy [1999]. The author described his stemming method as "quick and dirty," as it employs few morphological rules that could conflate singular and plural forms of nouns and adjectives. The method cannot handle variations in person, tense, or verb forms. The author proposed another light stemming method named "UniNE" in Savoy [2006] that uses 27 stemming rules. It could handle various inflectional variations and can also remove certain derivational suffixes. In addition to these light stemming approaches, an aggressive stemmer was also developed for the French language (included in Snowball standard stemmers) that stems word in six different steps [Porter 2001].

—*Spanish Stemming Methods*. For stemming in Spanish, Honrado et al. [2000] proposed a stemming method that uses two set of rules. The first set of rules stems regular verb forms using two dictionary lookup processes and the second set of rules stems irregular word forms that can involve up to three dictionary lookup processes. Figuerola et al. [2001] proposed another rule-based Spanish stemmer that uses finite-state automation for each suffix that expresses a number of rules in which the suffix can be attached to the root. The authors also studied the effect of an inflectional, derivational stemmer on the Spanish CLEF document set and reported that inflectional stemming is better than derivational in retrieval accuracy. A Spanish version of the Porter-style stemmer is also available on the Snowball website, which uses three steps to get the stem of the words.

—*Latin Stemming Methods*. The first rule-based stemmer for Latin is described in Schinke et al. [1996]. It removes or replaces noun and verb suffixes using a list containing 25 verb suffixes and 19 noun suffixes. It thus gives two different roots, one noun and another verb root. The stemmer is also incorporated into Snowball stemmers [Porter 2001]. But it did not gain much popularity as it cannot handle inflections in pronouns, adverbs, adjectives. Recently, Khoury et al. [2015] developed another rule-based stemmer that stems the word in three simple steps using the Wiktionary database provided by Wikimedia. The stemmer achieved an accuracy of 78% in finding the roots of the variant words.

—*Italian Stemming Methods.* For stemming in Italian, Monz and Rijke [2002] developed a lexical rule-based stemmer that removes inflections in number, case, and tense using a dictionary lookup mechanism. The author used the stemmer in an Information Retrieval task using the FlexIR system developed by the first author and reported an increase of 25% in Mean Average Precision. The Porter version of the Italian stemmer is also included in Snowball standard stemmers, which remove pronoun, verb, and other standard suffixes using the longest-match principle in four steps.

—*Portuguese Stemming Methods.* For stemming in Portuguese, we found the first rule-based stemmer proposed by Orengo and Huyck [2001], which removes various inflections in eight steps using 199 rules. Each step makes use of a different set of rules that are arranged and applied in a particular order. Only one rule is applied in each step. Each rule defines the ending to be removed, the minimum stem length, the string to be appended (optional), and a set of exceptions. The stemmer outperformed the Snowball Portuguese Stemmer [Porter 2001], which removed different suffixes in five steps. Alvares et al. [2005] proposed another rule-based stemmer, called STEMBER, which uses some of the rules described by Orengo and Huyck [2001]. It first handles special cases such as verbs with irregular conjugations and then removes prefixes and suffixes. The suffixes are removed on the longest-match principle using a set of 394 rules. Savoy [2006] also developed a rule-based Portuguese Stemmer that removes plural and feminine inflections in nouns and adjectives using 10 rules for plural forms and 13 rules for feminine forms. Soares et al. [2009] improved the Snowball Portuguese stemmer by exploring features of the language and experimentally proved that the improved version of the stemmer provides better stems in far less time.

4.2.2. Stemming in Slavic Languages. Slavic languages are predominantly spoken in Eastern and Western Europe and have fusional morphology.[1] The stemming methods have been developed for major languages like Polish, Russian, Czech, Bulgarian, Slovene, Croatian, and so on.

—*Polish Stemming Methods.* Polish belongs to the Slavic language family, which has a large number of inflectional rules, thereby making word variations quite common. SAM-95 [Szafran 1996] is the first morphological stemmer that makes use of a table of suffixes through which potential stems are generated and matched with a dictionary of root words. Another dictionary-based stemmer named Lametyzator has been proposed by Weiss [2003]. The dictionary contains a list of inflected word forms with the stem compressed into Finite State Automata (FSA), thereby reducing the size of the dictionary and making the stemming process fast. Białecki [2004] developed a rule-based Polish stemmer called Stempel that makes use of rules that adds, truncates, or substitutes characters to get the stem of the inflected words. The rules are stored as a set of patch commands. Wachnicka [2004] proposed another rule-based stemmer named REG, which uses simpler rules as compared to Stempel and are stored in human readable form. Weiss [2005] developed a hybrid stemmer by combining dictionary-based stemmer Lametyzator and rule-based stemmer Stempel. The word to be stemmed is first searched in Lametyzator and, if not found, it is stemmed using rules of Stempel.

—*Bulgarian Stemming Methods*. Nakov [2003] developed a Bulgarian stemmer named BULSTEM, which uses a morphologically rich dictionary containing nearly 889,665 word forms. In order to stem the word, the dictionary is scanned, and the various

[1] Fusional languages are a type of synthetic language, distinguished from agglutinative languages by their tendency to use a single morpheme to denote multiple grammatical, syntactic, or semantic changes.

matching rules along with their frequencies are gathered. The longest rule is then selected with a restriction that the resultant stem contains a vowel. Savoy and Berger [2006] also developed a rule-based stemmer that removes inflections and tackles many morphological difficulties of the Bulgarian language such that suffixes are not added to denote grammatical cases.

—*Slovene Stemming Methods.* Popovic and Willet [1992] developed a rule-based stemmer for Slovene, which uses a suffix list containing 5,276 suffixes with minimum stem constraint. Each suffix also contains a code for one of the eight context-sensitive rules that are applied if minimum stem condition is satisfied. After the removal of the suffix, three sets of recoding rules decide whether the stem needs to be recoded or not. The authors experimentally verified that the stemmer is quite effective in Information Retrieval tasks and concluded that with an increase in morphological complexity of the language, the efficiency of the stemmer increases.

—*Czech Stemming Methods.* For stemming in Czech, Dolamic and Savoy [2009a] proposed light and aggressive stemming approaches. The "light" stemmer removes inflections in nouns and adjectives using a set of 52 rules. The suffixes are also removed from the words belonging to other parts of speech but having the same suffix, thereby producing incorrect stems. The "aggressive" stemmer also removes certain frequently occurring derivational suffixes along with inflectional suffixes. The aggressive stemmer performed better than the light stemmer in IR tasks. Both the light and the aggressive stemmers are also included in Snowball stemmers in March 2012.

—*Russian Stemming Methods.* Dolamic and Savoy [2009b] developed "light" and "aggressive" stemming approaches for Russian using the same approach as for other European languages like Czech and French. The "light" stemmer removes inflectional endings in nouns and adjectives using 57 rules. Four more rules are used to normalize the stem. The aggressive stemmer mainly removes relational endings and adjectival qualitative. It makes use of 40 additional rules along with 57 rules in the light stemmer to handle both inflectional and frequently occurring derivational suffixes. A Porter-style Russian stemmer is also incorporated in Snowball standard stemmers that remove nouns, adjectives, verbs, and other inflections in four steps.

—*Croatian Stemming Methods.* Croatian, like other Slavic languages, has intense inflections. In our study, we found that the first rule-based stemmer for Croatian was proposed by Lauc et al. [1998]. It can handle only inflectional variations in the word forms without using a lexicon. The stemmer provided satisfactory results in retrieval experiments. Another rule-based stemmer was proposed by Ljubeši et al. [2007] that removes noun inflections using five sets of rules specifying different types of constraints. Šnajder et al. [2008] proposed a dictionary-based stemmer for Croatian that also removes inflectional variations in the word forms. It automatically creates an inflectional lexicon from the raw corpus, and the authors claimed that the same method applies to any other language with a similar morphology.

4.2.3. Stemming in Uralic Languages. Uralic languages are mainly spoken in Eastern, Central, and Northern Europe and are highly inflectional. We found rule-based stemming methods mainly developed for Hungarian. Savoy [2006] developed the "UniNE" stemmer for the Hungarian language as for other European languages like French, Portuguese, and German. The stemmer uses 17 rules for removal of possessive endings, 21 rules for removal of case markers, and 2 rules for plural removal. The light stemming procedure of the same stemmer removes only frequent suffixes using 13 rules. The Porter-style Hungarian stemmer included in Snowball standard stemmers [Porter 2001] removes inflections in nouns and other special and frequent suffixes in nine steps.

4.2.4. Stemming in Albanian. The Albanian language is a separate language branch and has shown no close similarity to any other language of the Indo-European family. From the viewpoint of computational linguistics, the Albanian language has not been much explored. In our study, we found that Karanikolas [2009] made the first attempt to develop a single-step rule-based stemming algorithm for Albanian. The proposed stemming method is based on suffix removal on the basis of the longest-match principle, and the suffix list for the language is derived using a set of five grammar books. The method is manually evaluated, and the author reported an accuracy of 80% in stemming the variant word forms. Sadiku [2011] proposed another rule-based stemmer that removes noun and verb inflections using a set of suffixes and language-specific rules. Biba and Gjati [2014] pointed out that earlier stemming works do not handle composite words that are quite common in the Albanian language. So, the authors developed a rule-based stemmer for composite words through an analysis of the morphology and the structure of words in the Albanian language. The authors tested the performance of the stemmer in text classification tasks using several classifiers and claimed that stemming of composite words significantly enhances the performance of classifiers.

4.2.5. Stemming in Hellenic Languages. Hellenic is another language branch belonging to the Indo-European family and includes only the Greek language. Greek is a fusional language in which a single morpheme is used to indicate multiple syntactic, semantic, and grammatical variations. Kalamboukis and Nikolaidis [1995] developed the first Greek rule-based suffix stripper method, called TZK, which uses three suffix lists for noun, verb, and adjective inflections. The stemmer removes both inflectional and derivational suffixes by checking the suffix lists according to the grammar category. The authors modified the proposed stemming method in their work published in Kalamboukis and Nikolaidis [1999] by improving the suffix tables and other errors. The authors tested both stemmers in IR tasks and concluded that the stemmers should also include corpus-specific information to avoid bad conflations. Ntais [2006] developed another suffix stripping stemming method that uses 166 suffixes for removing noun, verb and adjective suffixes, thereby outperforming TZK method. The stemmer uses 22 rule sets that remove suffixes on the basis of the longest-match principle. The major limitation of this method is that it can handle words in capital letters only. Adam et al. [2010] developed a hybrid approach for stemming that first determines the syntactic category of the word and then applies suffix stripping depending on the grammatical category of the word.

4.2.6. Stemming in Germanic Languages. The Germanic languages are mostly spoken in North America and Central, Western, and Northern Europe. Most of the stemming work in this language family is reported in Dutch, German, and Swedish.

—*Dutch Stemming Methods.* Kraaij and Pohlman [1994] developed the first rule-based stemmer that works like an English Porter stemmer. It removes inflectional, derivational, and frequently occurring affixes using a six-rule set. The rules in each group are mutually exclusive and ordered. Moreover, rules may contain additional context-sensitive conditions. Kraaij and Pohlman [1996] tested the proposed Porter-style stemmer along with other rule-based stemmers in Information Retrieval tasks. The results show that stemming inflectional variations give consistent improvement as compared to no stemming and full linguistic stemming. Gaustad et al. [2002] proposed a dictionary lookup-based stemmer that uses FSA-encoded dictionary information from Celex [Baayen et al. 1993]. It contains 1, 24, 136 root words and 3, 81, 292 variant word forms for Dutch. The proposed stemmer reported an accuracy of 98% in text classification tasks.

—*Swedish Stemming Methods.* Carlberger et al. [2001] developed a rule-based stemmer for Swedish that makes use of a set of suffix removal rules (nearly 150) that are applied to obtain the stem in four steps. In each step, one rule is applied, and each rule contains a pattern to be matched with the ending of the word along with a set of constraints. The stemmer showed significant improvements in retrieval accuracy.

—*German Stemming Methods.* For stemming in German, a Porter-style rule-based stemmer has been included in standard Snowball stemmers [Porter 2001]. The rules in the stemmer contain information of endings to be removed, new strings to be appended, length of the word, and special conditions. The suffixes for the language are identified through their frequencies in a large list of words. Braschler and Ripplinger [2004] studied the effect of decompounding and stemming of words using six different stemmers, including German Porter Stemmer and Linguistica [Goldsmith 2001] in Information Retrieval. The authors found that even a simple stemming approach and decompounding of words can significantly improve the retrieval performance. Savoy [2006] developed a German rule-based stemmer that uses 11 rules to remove various grammatical and plural variations.

4.3. Rule-Based Stemmers for Asian Languages

Like European languages, rule-based stemmers have also been developed for Asian languages using various linguistic resources and language-specific rules. In this subsection, we describe rule-based stemmers for 13 Asian languages belonging to four different language families, namely Indo-Aryan, Iranian, Dravidian, and Semitic.

4.3.1. Stemming in Indo-Aryan Languages. Indo-Aryan languages are mainly spoken in South Asia, and stemming methods have been developed for many major languages of the family, namely Hindi, Bengali, Marathi, Gujarati, Punjabi, Urdu, and Assamese.

—*Hindi Stemming Methods.* For stemming in Hindi, we found that Larkey et al. [2003] developed the first suffix stripping-based stemming method (UMass). It removes inflectional variations in number, gender, tense, and normalization on the basis of longest match first using a list of 27 suffixes. In the same year, Ramanathan and Rao [2003] also developed a suffix stripper that removes inflectional variations using an extensive suffix list containing 65 suffixes. Shrivastava et al. [2005, 2008] developed a hybrid stemming method that first checks the word to be stemmed in the lexicon. If it is found, then the word is returned as the stem itself. Otherwise the matching suffix is removed or replaced with a different set of rules. Dolamic and Savoy [2010] developed a "light" and "aggressive" stemming method. The "light" stemmer removes variations in nouns and adjectives using 20 rules, whereas the "aggressive" stemmer uses 49 rules to remove some derivational suffixes along with inflectional variations. Mishra and Prakash [2012] developed a hybrid stemmer named MAULIK, which combines brute-force and suffix stripping methods. The stem of the word is first searched in the lexicon and, if not found, then the word is stemmed using suffix stripping rules. Gupta [2014] developed another rule-based stemmer that handles variations in nouns and achieved an accuracy of 83% in stemming variant words.

—*Bengali Stemming Methods.* Islam et al. [2007] proposed a "light" rule-based stemmer that removes noun, verb and adjective inflections in Bangla using a suffix list. The stemmer is also customized to be used in a spell-checking routine. Dolamic and Savoy [2010] proposed "light" and "aggressive" stemming method for Bengali and evaluated them in IR tasks. The "light" stemmer removes inflectional variations using 70 rules, and the "aggressive" stemmer removes both inflectional and few derivational suffixes using 85 rules. The "aggressive" stemmer performed better than the "light" stemmer in IR experiments. Das and Bandyopadhyay [2010] developed two stemming approaches that cluster morphological variants. In the first approach, the suffixes are stripped using a suffix list sorted according to the length of suffixes and, in the second approach, a minimum edit distance methodology is combined with suffix stripping. Another rule-based stemmer has been proposed for Bengali by Mahmud et al. [2014] that removes inflections algorithmically in a number of steps. The stemmer achieved an accuracy of 88% in stemming the variant word forms.

—*Marathi Stemming Methods.* Dolamic and Savoy [2010] developed "light" and "aggressive" stemming methods for Marathi also as for Hindi and Bengali. The "light" stemmer makes use of 51 rules to remove most commonly occurring suffixes in nouns and adjectives, and the "aggressive" stemmer uses 82 rules to remove inflectional and most frequent derivational suffixes. Like Hindi and Bengali, "aggressive" stemmer performed better than inflectional in retrieval experiments.

—*Punjabi Stemming Methods.* For stemming in Punjabi, we found two rule-based stemmers. The first stemmer is described in Kumar and Rana [2011], which uses a brute-force approach. The word to be stemmed is searched in the lookup table, which contains relations between the variant words and their root forms. Another stemmer proposed by Gupta and Lehal [2011] uses an algorithmic approach to remove inflections in nouns and proper names. The accuracy of noun and proper name stemmer is reported to be 87%.

—*Gujarati Stemming Methods*. Patel et al. [2010] proposed a hybrid stemming approach that combines the Goldsmith [2001] optimal split point approach with manually formed suffix list. The authors reported that combination of both approaches boosted the performance by 17%. Ameta et al. [2012] proposed a rule-based stemmer for Gujarati, which removes noun, verb, adjective, and adverb inflections using suffix list of 167 suffixes.

—*Assamese Stemming Methods.* For stemming in Assamese, we found only one rule-based stemmer, which was proposed by Saharia et al. [2012]. The proposed stemmer first makes use of a rule engine that generates a list of suffixes and then employs a suffix stripping operation to stem the word. The accuracy of the suffix stripping method is reported to be 82%.

—*Urdu Stemming Methods.* Akram et al. [2010] developed an Urdu stemmer (Assas-Band). It first deletes the prefix and postfix using two exception lists and then adds character(s) to obtain the base word form. The suffix stripping operation is bypassed if the root of the variant word form is successfully found in the lookup table. Kansal et al. [2012] developed another rule-based stemmer for Urdu. The authors first generated a list of affixes from a corpus and stored them according to their frequency and then used this list to remove the affixes from the variant words.

4.3.2. Stemming in Dravidian Languages.

Dravidian languages are also spoken mainly in Southern Asia. Most of the stemming work in this family is reported for Tamil and Kannada.

—*Tamil Stemming Methods.* Ramachandran and Krishnamurthi [2012] developed an iterative rule-based stemmer for Tamil. The stemmer can handle inflectional variations in suffixes but cannot remove variations in prefixes. The accuracy of stemmer is reported to be 84%. Lushanthan et al. [2014] developed a morphological stemmer for Tamil that uses a lexicon and morphological rules to return stems that are real words and not truncated forms. The rules of stemming are expressed as regular expressions using finite-state methods.

—*Kannada Stemming Methods.* Hegde et al. [2013] developed a suffix stripping stemming method for Kannada that uses 14 major classes of suffixes of Kannada. It removes inflectional variations in nouns, verbs, adjectives, and articles. Deepamala and Kumar [2014] developed a hybrid stemmer that combines brute-force and suffix stripping approaches. The word to be stemmed is first searched in a stem-list lexicon containing 18,804 root words and, if not found, then a suffix stripping operation is applied. The stemmer gave promising results in the classification of Kannada documents.

4.3.3. Stemming in Iranian Languages. Iranian languages are mainly spoken in Southwest, Central, and South Asia. The main stemming work in this family is reported for Kurdish, Persian, and Pashto.

—*Kurdish Stemming Methods*. In Kurdish, we found only one rule-based stemmer named Jedar, described in Salavati et al. [2013]. It recursively removes multiple suffixes attached to Kurdish root words. The suffix removal rules are stored and applied in a definite order, and each rule is associated with a certain context-sensitive condition. The authors compared their rule-based approach with the statistical stemmer GRAS [Paik et al. 2011a] and reported that both perform well in IR tasks for Kurdish.
—*Pashto Stemming Methods*. For stemming in Pashto, Aslamzai and Saad [2015] developed a morphological stemmer that removes inflectional and derivational variations in Pashto affixes. The returned stems are meaningful words that can be found in the dictionary. The proposed method is manually tested by two native speakers and achieved an accuracy of 87%.
—*Persian Stemming Methods*. Harmanani et al. [2006] developed a rule-based suffix stripping method for Persian as for Arabic and Hebrew. It removes affixes in the Persian language using a set of rules interpreted by a rule engine. The rule engine can be adapted to any other language by just modifying the language-specific rules. Sharifloo et al. [2008] developed a high-precision rule-based bottom-up approach for stemming. It works in three phases to obtain the stem of the word. The proposed method gave promising results in stemming and flexibility. Estahbanati et al. [2011] also proposed a multi-phase rule-based algorithm. It removes plural noun endings, possessive noun endings, verb endings, and few other frequently occurring endings using a suffix list on the basis of longest match.

4.3.4. Stemming in Semitic Languages. Semitic languages are mainly spoken in Western Asia and parts of Africa. Most of the stemming work in this language branch is reported for the Arabic language. Arabic has a rich system of inflectional variation. The variations are usually done by adding affixes, infixes, double consonants, or vowels. A number of light (stem-based) and aggressive (root-based) Arabic stemming techniques have been proposed in the literature. Light stemming methods remove prefixes and suffixes, whereas aggressive stemming methods can also handle infixes. Among root-based approaches, Khoja and Garside [1999] proposed the first Arabic rule-based stemmer (Khoja stemmer), which deletes infixes from variant word forms and then compares it with a list of patterns to provide the root. A number of improvements have been proposed [Taghva et al. 2005; Kchaou and Kanoun 2008; Al-Kabi 2013] in the Khoja stemmer that removes problems like handicapped stems, missing patterns, and so on.

Among stem-based methods, Larkey et al. [2002, 2007] developed five light stemming approaches with minor variations, namely Light1, Light2, Light3, Light8, and Light10. These stemmers delete various suffixes, definite articles, and conjunctions (prefixes) to obtain the stem. In retrieval experiments, the performance of Light10 is reported to be the best. Chen and Gey [2002] proposed the Berkley stemmer, which elaborates on the context-sensitive constraints and suffix/prefix list of the Light10 stemmer. Harmanani [2006] proposed an extensible rule-based stemmer that removes affixes using a rule engine. The stemmer is fully tested on an Arabic test collection, and it achieved significant improvements. El-Beltagy and Rafea [2011] developed another rule-based stemmer that generated stem tables from a large Arabic text collection and can handle the case of broken plurals[2] along with affix removal. Elrajubi [2013] also proposed a light stemming approach that employs a recoding method after affix removal.

5. STATISTICAL STEMMING METHODS

Statistical stemmers are based on unsupervised or semi-supervised learning of the morphology of a language from an ambient corpus. Due to the feature of language independence, the scope of these techniques is wide, as they can be applied to the corpus of many languages where other linguistic resources are incomplete. Moreover, these techniques show features of the ambient corpus and in some cases have been found to be more advantageous as compared to rule-based stemmers [Paik et al. 2011b]. In the subsequent subsections, we describe various lexical analyses, corpus analyses, and character n-gram-based statistical stemming methods proposed in the literature.

5.1. Lexicon Analysis-Based Methods

5.1.1. Stemming Using Frequency of Substrings. These methods identify potential suffixes and stems of the words in the corpus on the basis of the frequency of substrings. Researchers have used these frequency counts in different ways to discover stems of the variant words.

Hafer and Weiss [1974] first used the count of the variety of letters of the successors and/or predecessors to decide whether the word should be divided or not. The idea behind calculating the successor letter variety count is that, in a word, the letter at any position is dependent on the letters preceding it, and the dependence increases as we move towards the word. But in the case of long words, the value of these counts is so small near the end of the word that it does not provide any information. So the test word is reversed, and the predecessor letter variety count is also determined. Once the successor and predecessor letter distribution count is available, then one technique or a combination of the following techniques is used to partition the word: threshold values, predecessor/successor entropy, count being on peak or plateau, and the prefix/suffix being a complete word in the corpus. The authors performed experiments using 15 different combinations and reported that best precision and recall values are obtained using a combination of cutoff values and entropy values. Stein and Potthast [2007] also used an alternate form of the letter successor variety to develop an unsupervised stemming algorithm. It uses tree structure suffixes along with some pruning rules to find the successor variety of internal nodes of the tree.

Goldsmith [2001] also proposed a method to discover stems and derivations using the calculation of substring frequency and the Minimum Description Length (MDL) approach. It first calculates the frequency of stem and derivation at each point of the word and then finds the best split of the word. The best split of the word is that which minimizes the total compressed length of the corpus, and every occurrence of the word in the corpus has the same split. The segmentation of the words into stems and suffixes are organized into signatures (a data structure that groups all stems that share a common set of suffixes). The proposed method can handle out-of-vocabulary words by identifying the cases where one signature's suffixes are a proper subset of another's [Chan 2006]. For example, consider two signatures learned from an English corpus: "Signature (1) with suffixes {e, ed, es, ing} and stems *achiev, chang, charg*, etc." and "Signature (2) with suffixes {e, ed, er, es, ing} and stems *advertis, announc, bak,* etc." Since the suffix set of signature 1 is a proper subset of suffix set of signature 2, the suffixes of signature 2 can also be attached to the stems of signature 1. Hence, the word "*achiever*" will be stemmed to "*achiev*" even if it is not present in the corpus.

[2] In Arabic, broken plurals are formed by changing the pattern of vowels and consonants inside the singular form.

The major limitation of this method is that it is very slow as it takes days to get trained for a medium-sized corpus. The framework of this morphological analyzer, named Linguistica, is freely available [Goldsmith 2006].

Melucci and Orio [2003] also used frequency of characters and substrings with Hidden Markov Model (HMM) for unsupervised learning of stemming rules. In this approach, the letters of the words are represented as the states of HMMs. These states are divided into two sets: the prefix set and the suffix set with some presumptions. There are multiple paths or HMMs that correspond to a single word. For any word, the best HMM is chosen using an estimation of probabilities that depends on the frequency of substrings. In the chosen path, the change from the state that belongs to the prefix set to the state that belongs to the suffix set is termed as the break point, and all the letters before this breakpoint represent the stem of the word. In IR experiments, the proposed method performed comparably to the Porter stemmer in English and four European languages, namely French, Dutch, Italian, and Spanish.

Bacchin et al. [2005] proposed a method to discover stems and derivations using a frequency of substrings and link analysis method. The proposed method is an extension of the authors' work in Bacchin et al. [2002] as the mutual relationship between the stems and suffixes is modeled using probabilistic interpretation. The stemming method first splits the words at every possible point and forms sets of substrings. These sets are then used to determine the prefix and suffix scores using directed graphs and a Hyperlink Induced Topic Search (HITS) link analysis algorithm [Kleinberg 1999]. The best split is then determined on the basis of the maximum probability of a suffix-prefix pair.

5.1.2. Suffix Discovery Using Frequency of Suffixes. Another approach used in unsupervised stemming is based on identifying suffixes from the corpus on the basis of their frequency. The identified suffixes are then used in the process of suffix stripping to obtain the stem of the words.

Oard et al. [2001] developed a method to identify suffixes based on their frequencies from first 5,000,000 words in the corpus. The frequencies of suffixes of length one to four characters is computed from these words. In order to remove the effect of partial suffixes ("-ng" is part of "-ing"), the suffix frequencies are modified by subtracting the count of the most frequent subsuming ending of next longer length from each ending. In order to decide number of suffixes of each length, the rank at which the second derivative in the plot of count and rank is maximized is chosen as threshold. In order to stem the word, the first matching suffix in the list (from the top) is removed. The proposed method gave good results in French, but in German and Italian the results were not so promising.

Paik and Parui [2011] also proposed a method based on suffix discovery from the lexicon. The proposed method finds equivalence classes of variant words using potential suffixes (frequently occurring suffixes) and common prefix knowledge. The words are first divided into different classes according to the common prefixes of specific length and then the strength of various classes is calculated using the potential suffix information. If the strength is good enough, then it is considered as the root of the class. Otherwise, another root from the class is found iteratively. The proposed method provided promising results in IR experiments for English and highly inflectional languages like Marathi and Hungarian. Yadav et al. [2012] also applied potential suffix information to discover suffixes from the lexicon. The algorithm is tested on six Asian languages, namely Hindi, Marathi, Gujarati, Bengali, Tamil, and Odia, and performed well in all the languages.

5.1.3. Word Conflation Using String Similarity Measures. Majumder et al. [2007] proposed a method to group morphological variants of words on the basis of a string similarity measure named Yet Another Suffix Stripper (YASS). The authors proposed four string similarity metrics that support the longest-match principle and avoid early mismatching. The authors experimentally verified that measure given in Equation (1) is most effective and used it in their stemming algorithm,

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

where n is the length of the longer string and m is the point of the first mismatch. Once the string similarity is calculated, then the clusters are obtained using a graph-based complete linkage technique. In this clustering, the similarity of the groups (clusters) is calculated as the minimum similarity of any element of the first group and any element of the second group. The author tested the stemmer in IR tasks for English, French, and Bengali and reported significant improvements. The stemmer was later also tested on Hungarian and Czech collections [Majumder et al. 2008], and promising results have been obtained. Fernandez et al. [2011] also proposed a method of word conflations using extended edit distance.

5.1.4. Graph-Based Clustering. Paik et al. [2011a] proposed an unsupervised stemming method that applies knowledge of suffix pairs to the undirected graphs. The proposed method first groups the words of the lexicon into classes according to common prefix length and then finds suffix pairs in each class. A weighted undirected graph is then constructed whose vertices are the distinct words of the corpus, and there is an edge between words u and v if they induce a frequent suffix pair. The weight of the edge between u and v is the frequency of the suffix pair. The graph is then decomposed to generate various classes of related words using a metric called cohesion. The proposed stemmer outperformed in retrieval experiments against a number of statistical stemmers [Oard et al. 2001; Goldsmith 2001; Majumder et al. 2007] for various European and Asian languages.

Table II compares various lexicon analysis-based stemmers in terms of their advantages, disadvantages, strength (mean number of words per conflation class), and experimental results.

5.2. Corpus Analysis-Based Approaches

5.2.1. Word Clustering Using Co-occurrence of Words. These stemming methods conflate morphological variant words based on co-occurrence information of words in a document or a community. Xu and Croft [1998] proposed a stemming approach that uses co-occurrence data. The authors highlighted that the conflations based on corpus are better than conflations produced by rule-based stemmers because words have different meanings in different contexts. For instance, the word “diamond” can be used as a precious gem, a geometrical shape, or a baseball ground. So its occurrence in the corpus can clarify the context in which the word is used. The method first computes co-occurrence between word pairs in a 100-word window using a variation of the Expected Mutual Information measure and then groups words using a graph-based connected component and/or optimal partitioning clustering methods.

Paik et al. [2011b] pointed out that the stemmer proposed by Xu and Croft [1998] has certain shortcomings like corpus-dependent parameters and the graph-based clustering algorithm produce long equivalence classes. So the authors proposed a novel graph-based nearest-neighbor clustering algorithm that does not require any parameter tuning. It first groups words into different classes according to common prefix of specific length and then computes co-occurrence between the words in the same class using a metric that only depends on the frequencies of the word in the document. The co-occurrence strength is then used by the graph clustering algorithm to form equivalence classes. The stemmer performed better than stemmers by Xu and Croft [1998]

Table II. Comparison of Lexicon Analysis-Based Stemmers

Reference	Technique	Advantages	Disadvantages	Stemmer Strength	Experimental Results
Hafer and Weiss [1974]	Frequency of letter successor and/or predecessor variety	Accurate stems are returned due to the use of structure linguistic, fast and effective for the large corpus.	A number of corpus dependent parameters.	Moderate	In IR, performance is comparable to traditional methods. Stemming accuracy of 61% in ADI and 74% in CPC collections
Oard et al. [2001]	Suffix Discovery Based on frequency.	Computationally simple and fast	Ignores certain infrequent suffixes.	Light	Nearly 50% improvement in French IR experiments but no improvement in English, German, and Italian.
Goldsmith [2001]	Frequency of sub-string and MDL approach.	High out-of-vocabulary rate, no parameter tuning.	A Large number of computations and very slow.	Moderate	Correctly stemmed 83% of English and French words.
Melucci and Orio [2003]	Frequency of substring applied to HMMs	The best path is synergy between modeling of prefixes and suffixes.	Tends to over-stem words, Complex.	Moderate	In IR tasks, performance is comparable to Porter in English and four European languages.
Bacchin et al. [2005]	Frequency of substrings and link analysis method	Fully automated, requires no parameter tuning	Computationally Intensive	Moderate	In IR tasks, performance equivalent to Porter for English and four European languages.
Majumder et al. [2007] (YASS)	Clustering based on string similarity	A large number of words per conflation class.	Corpus dependent parameters, cannot handle long suffixes	Strong	In IR tasks, significant (up to 15%) improvement in MAP for three languages.
Paik and Parui [2011]	Suffix discovery and common prefix knowledge	Fast, appropriate parameter selection method.	Ignores some infrequent suffixes	Light	In IR, large improvements (35%–45%) in Marathi and Hungarian but small improvements in Bengali & English (8%–18%).
Paik et al. [2011a] (GRAS)	Suffix pair knowledge applied to graphs	Fast and provides large number of words per class	Over-stemming in some cases, corpus dependent parameters	Strong	In IR tasks, large improvements (up to 60%) in MAP for seven Asian & European Languages.

and Majumder et al. [2007] in Information Retrieval tasks for four European and two Asian languages.

Paik et al. [2013] proposed another stemming method wherein a new metric based on random co-occurrence of words is used to find an association among word pairs. The co-occurrence information and structural knowledge of words are then used to group morphologically related words. The stemming method is further customized to perform query-based stemming that provides those variants of the words that are related to the original context of the query terms. It is done to avoid any bad conflations produced due to repeated occurrence of the words in the corpus that are not morphologically related. Both the query-independent and query-dependent methods performed well in Information retrieval tasks for English and non-English languages.

5.2.2. Word Clustering Using Distributional Similarity. Distribution similarity is a more advanced form of co-occurrence information that can be obtained from the corpus. Bhamidipati and Pal [2007] first used this metric to refine equivalence classes of words in the corpus. The proposed technique is based on the assumption that various documents belong to different categories in the corpus, and the words are multinomial distributed among these categories. So the count of occurrence of these words in various categories is used to find distribution similarity among the words. The authors refined the equivalence classes formed by truncation and the Porter stemming methods and reported that refinement of these classes leads to improvement in retrieval and classification performance.

Peng et al. [2007] also used distribution similarity of words to develop a context-sensitive stemmer for web search. The authors pointed that blind stemming in web search has few limitations. First, it converts the various terms of the query without considering the context in which the term is used. Second, traditional stemming performs blind comparisons of all the occurrences in the documents. So he proposed an approach that performs context-sensitive stemming at the query and document sides. A context-based analysis is performed during query expansion that decides which terms must be expanded with which morphological variation to get the best results. According to the expanded terms of the query, a conservative context-based comparison is performed at the document side.

5.2.3. Word Clustering Using Lexical and Semantic Features. Brychcin and Konopik [2015] proposed an unsupervised stemming method that uses both lexical and semantic information from the corpus. The stemmer works in two steps. In the first step, modified Minimal Mutual Information (MMI) clustering is used to group words that are lexically and semantically related (having the same semantics and share a common prefix). These clusters are used as training data for a maximum entropy classifier that encodes context-specific stemming rules into features. The authors tested the proposed method in three different types of experiments: Inflection Removal, Language Modeling, and Information Retrieval. The stemmer performed well in all three experiments and is hence used as a multi-purpose tool.

Table III compares various corpus analysis-based stemmers in terms of their advantages, disadvantages, and experimental results.

5.3. Character *N*-Gram-Based Approaches

McNamee and Mayfield [2004] first used character n-grams to develop the unsupervised stemming method. In the character n-gram stemming technique, n-gram (n-adjacent characters) are selected from the various words in the corpus. The root words that are unique will have less frequency, whereas the morphological variants that is, suffixes or prefixes, will have a comparatively higher frequency. Usually, the size of n is taken as 4 or 5. This technique will have a large number of indexes per word as

Table III. Comparison of Corpus Analysis-Based Stemmers

Reference	Technique	Advantages	Disadvantages	Experimental Results
Xu and Croft [1998]	Word clustering using EMIM based co-occurrence metric	Use corpus-specific information to avoid any bad conflation.	Corpus dependent parameters produce long clusters.	In English and Spanish IR, precision improved by 10%.
Bhamidipati and Pal [2007]	Word clustering based on distribution similarity	Improves incorrect conflations using advanced co-occurrence data.	Can only refine classes already formed by a stemmer.	Improved retrieval precision and recall by 25%. Achieved better classification accuracy as compared to no stemming.
Peng et al. [2007]	Context-sensitive stemming using distributional similarity	Performs context based stemming at query and document side in web search.	Not fully automated and in some cases perform under-stemming	Significant improvement in DCG and overall query traffic.
Paik et al. [2011b]	Nearest neighbor clustering using term frequency based co-occurrence metric	No manual parameter tuning, nearest neighbor clustering produce improved clusters.	Involves lot of graph related computations.	20%–50% improvements in retrieval performance for various English, European and Asian languages.
Paik et al. [2013]	Clustering based on co-occurrence, structural and query specific knowledge.	Provides variant words related to the query intent.	Computationally intensive	50% improvements in retrieval accuracy for Czech and up to 30% improvement in Asian languages.
Brychcin and Konopik [2015]	Word clustering using lexical and semantic features.	Requires small training data, high out of vocabulary rate	Decrease recall at the expense of precision	Performed consistently well in Inflection removal, Language Modeling & IR.

compared to other techniques and hence consumes a large amount of storage space. Though many other methodologies are suggested for reducing the size of the indices, they do not decrease the time of execution of the query.

Mayfield and McNamee [2003] also proposed a pseudo stemming approach. In this technique, the n-gram is chosen from the unique part that is, the morphological variant of the word. The inner n-gram of the word that has the maximum Inverse Document Frequency (IDF) is chosen as a single pseudo stem for the word. The high value of IDF is so chosen because the variants of the word will appear in many words in the document and will have a low value of IDF. Ahmed and Nürnberger [2009] proposed a variation of n-gram for the Arabic language. The authors pointed out that a pure n-gram technique does not consider the sequence in which the n-grams occur. It may sometimes lead to large similarity among the words that are not morphologically related. So they modified the n-gram approach by considering the n-grams from certain definite locations according to their sequence. Jordan et al. [2006] also used n-grams for developing a stemming method called Swordfish. The proposed method finds potential stems and suffixes using joint probabilities of various n-grams of the words occurring in the corpus.

6. PERFORMANCE OF STEMMERS IN INFORMATION RETRIEVAL EXPERIMENTS

In this section, we present performance of several rule-based and statistical stemmers in ad hoc retrieval tasks on the basis of careful analysis of results in the literature. We considered a total of 16 state-of-the-art strong stemmers in our analysis of which 6 belong to each rule-based and lexicon analysis-based category and 4 belong to corpus

Table IV. Description of Stemming Methods and Abbreviations Used in Analysis

CATEGORY	ABBREVIATION	MEANING & REFERENCE
Rule-Based Stemmers	LOVIN	Rule Based Lovins Stemmer for English [Lovins 1968]
	PORTER	Rule Based Porter Stemmer for English [Porter 1980]
	RB-FR	Snowball French Stemmer (Porter Style) (http://www.snowball.tartarus.org/algorithms/french/stemmer.html)
	RB-BU	Rule Based Stemmer for Bulgarian [Savoy and Berger 2006]
	RB-BE	Rule Based Stemmer for Bengali [Dolamic and Savoy 2010]
	RB-MR	Rule Based Stemmer for Marathi [Dolamic and Savoy 2010]
Lexicon Analysis-Based Stemmers	OARD	Suffix Frequency Based Stemmer [Oard 2001]
	4-GRAM	Character n-gram Based Stemmer [Melucci and Orio 2003]
	LING	MDL Based Morphological Analyser (Linguistica framework) [Goldsmith 2006]
	YASS	Yet Another Suffix Stripper [Majumder et al. 2007]
	GRAS	Graph-Based Stemmer [Paik et al. 2011a]
	FCB	Fast Corpus-Based Stemmer [Paik and Parui 2011]
Corpus Analysis-Based Stemmers	XU	Corpus-Based Stemming using Co-occurrence [Xu and Croft 1998]
	SNS	Co-occurrence Based word Clustering [Paik et al. 2011b]
	QBS	Query Based Stemming [Paik et al. 2013]
	HPS	High Precision Stemmer [Brychcin and Konopik 2015]

analysis-based category. The complete description of stemmers along with the abbreviations used in the analysis is presented in Table IV.

6.1. Experimental System and Evaluation Metrics

The retrieval experiments are performed using the open-source TERRIER Information Retrieval System [Ounis et al. 2006]. It has been developed in Java by the School of Computing Science, University of Glasgow (available at http://terrier.org). It employs the Unicode Transformation Format (UTF) encoding scheme and provides indexing and searching support for many languages. The Inverse term Frequency with Bernoulli after-effect (IFB2) term weighting model [Amati and Rijsbergen 2002] is used, as it is suggested to be one of the best-performing models. The IFB2 model belongs to the Divergence from Randomness (DFR) paradigm and uses inverse term frequency as basic randomness model, with normalization factor B (Bernoulli after-effect) and with the presumption of hypothesis H2.

MAP is used as the primary evaluation metric in the experiments. Along with MAP, we also reported total number of relevant documents retrieved by each stemmer. In each table, the values in brackets denote relative improvements as compared to no stemming.

6.2. Description of Data

The retrieval experiments are performed on standard Text REtrieval Conference (TREC), Cross Language Evaluation Forum (CLEF), and Forum for Information Retrieval Evaluation (FIRE) document collection and query sets. In order to have a deeper analysis of the stemming performance, we considered two European (French and Bulgarian) and two Asian (Bengali and Marathi) languages besides English. The detailed description of corpora of all languages used in the experiments is given in Table V. The English experiments are performed on two TREC document collection of varying size. The first collection includes *Wall Street Journal* documents from TREC disks 1 and 2 along with 200 topics (1–200) from the ad hoc track. The second collection includes documents from *Financial Times*, Foreign Broadcast Information Service,

Table V. Summary of Test Corpora Used in the Analysis

Language	Source	#Documents	#Queries
English (Test Collection I)	TREC disks 1 and 2	1, 73, 252	200 (1–200)
English (Test Collection II)	TREC disks 4 and 5	5, 28, 155	100 (601–700)
French	CLEF 2005 & 2006	1, 77, 452	100
Bulgarian	CLEF 2006 & 2007	87, 281	100
Bengali	FIRE 2010	1, 23, 047	100
Marathi	FIRE 2008	99, 362	88

Table VI. Information Retrieval Results for English (Test Collection I)

METHOD	NO STEM	PORTER	OARD	4-GRAM	LING	YASS	FCB	HPS
MAP	0.272	0.295 (8.4)	0.295 (8.4)	0.268 (−1.5)	0.283 (4.0)	0.289 (6.2)	0.295 (8.4)	0.301 (10.7)
RELEVANT DOCUMENTS	13576	14161 (4.3)	14216 (4.7)	13402 (−1.3)	14008 (3.2)	14204 (4.6)	14363 (5.8)	14360 (5.8)

Table VII. Information Retrieval Results for English (Test Collection II)

METHOD	NO STEM	LOVIN	PORTER	XU	YASS	GRAS	SNS	QBS
MAP	0.268	0.273 (1.9)	0.280 (4.5)	0.283 (5.6)	0.285 (6.4)	0.282 (5.2)	0.283 (5.6)	0.314 (17.0)
RELEVANT DOCUMENTS	2784	2812 (1.0)	2910 (4.5)	2884 (3.6)	2890 (3.8)	2900 (4.2)	2888 (3.7)	3054 (9.7)

and *Los Angeles Times* along with topics 601–750. French experiments are performed on the CLEF 2005 and 2006 collections and for Bulgarian the CLEF 2006 and 2007 collections are used. The Bengali and Marathi experiments are performed on FIRE 2010 and FIRE 2008 document collection and topics, respectively.

6.3. Experimental Results

English Results. The Information Retrieval results for various stemmers examined on English Test Collection I (*Wall Street Journal* documents) for TREC 1–200 topics are shown in Table VI. It is clear from the table that various stemmers (except 4-GRAM) performed better than no stemming and improved Mean Average Precision from nearly 4% to 11%. HPS reported maximum improvement of 10.7% in MAP. PORTER, OARD, and FCB performed almost equal and showed an improvement of nearly 8% in MAP as compared to no stemming. YASS and LING showed small improvements of 6.2% and 4% respectively in MAP. 4-GRAM performed relatively poorer and decreased MAP by 1.5% as compared to no stemming.

The retrieval results on English Test Collection II (FBIS, FT, and *LA Times* documents) for TREC topics (601–700) are displayed in Table VII. In this collection, various stemmers also showed a small improvement in MAP (2%–17%) as in collection I. The QBS stemmer performed much better than the other stemmers and reported an increase of 17% in MAP. YASS, PORTER, XU, GRAS, and SNS performed nearly equally and showed an improvement of 4.5% to 6.5% in MAP as compared to no stemming. LOVIN performed relatively poorly and reported a small improvement of nearly 2% in MAP as compared to no stemming. Figures 3 and 4 display the comparison in terms of MAP values among various stemmers tested on English collections.

European Languages Results. Tables VIII and IX list the retrieval results of different stemmers examined on Bulgarian test collections (CLEF 2006, 2007) and French test collections (CLEF 2005, 2006), respectively. Bulgarian, being a morphologically rich

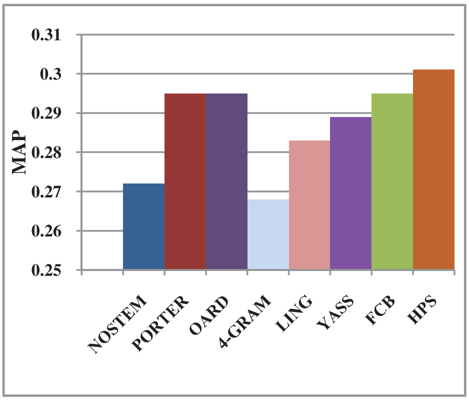

Fig. 3. MAP values for English Test Collection I.

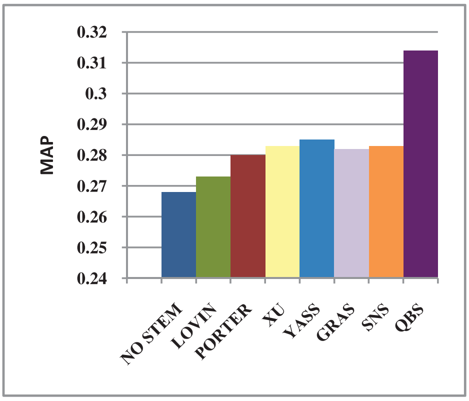

Fig. 4. MAP values for English Test Collection II.

Table VIII. Information Retrieval Results for Bulgarian (CLEF 2006–2007 Collections)

METHOD	NO STEM	RB-BU	XU	OARD	LING	YASS	GRAS	SNS
MAP	0.217	0.279 (28.6)	0.245 (13.0)	0.285 (31.3)	0.288 (32.7)	0.307 (41.5)	0.326 (50.2)	0.326 (50.2)
RELEVANT DOCUMENTS	1611	2003 (24.3)	1790 (11.1)	1944 (20.7)	1973 (22.5)	2085 (29.4)	2110 (31.0)	2101 (30.4)

Table IX. Information Retrieval Results for French (CLEF 2005–2006 Collections)

METHOD	NO STEM	RB-FR	XU	OARD	LING	YASS	GRAS	SNS
MAP	0.339	0.382 (12.7)	0.350 (3.2)	0.374 (10.3)	0.349 (2.9)	0.374 (10.3)	0.387 (14.2)	0.372 (9.7)
RELEVANT DOCUMENTS	3796	4102 (8.1)	3859 (1.7)	4061 (7.0)	3990 (5.1)	4055 (6.8)	4078 (7.4)	4067 (7.1)

language, showed large improvements in MAP (13%–50%) as compared to unstemmed words. GRAS and SNS reported a maximum increase of 50.2% in MAP followed by YASS, which improved MAP by 41.5%. The rule-based Bulgarian stemmer also showed a consistent increase of 28.6% in MAP. OARD and LING performed nearly equally and increased MAP by nearly 32%. XU performed relatively inferiorly to the other stemmers and showed an increase of 13% in MAP as compared to no stemming.

In the case of the French language, stemming is found to be less effective as compared to the Bulgarian language and showed small improvements in MAP ranging from 3% to 14%. GRAS showed a maximum improvement of 14.2% in MAP followed by the rule-based French Stemmer, which improved MAP by 12.7% as compared to no stemming. SNS, YASS, and OARD performed almost equally and reported an increase of nearly 10% in MAP. Like Bulgarian, the XU stemmer also performed relatively poorly in French and showed a small increase of 3% as compared to no stemming. Figures 5 and 6 compare the MAP values of various stemmers for Bulgarian and French, respectively.

Asian Languages Results. The retrieval results of various stemmers tested on Bengali (FIRE 2010) and Marathi (FIRE 2008) are displayed in Tables X and XI, respectively. In the case of the Bengali language, GRAS reported a maximum increase of 21.1% in MAP as compared to unstemmed words. SNS, YASS, and HPS performed consistently well and improved MAP from nearly 17% to 19%. The PORTER and XU

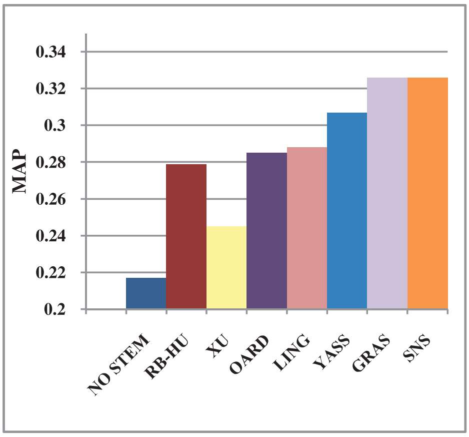

Fig. 5. MAP values for Bulgarian Test Collection (CLEF 2006, 2007).

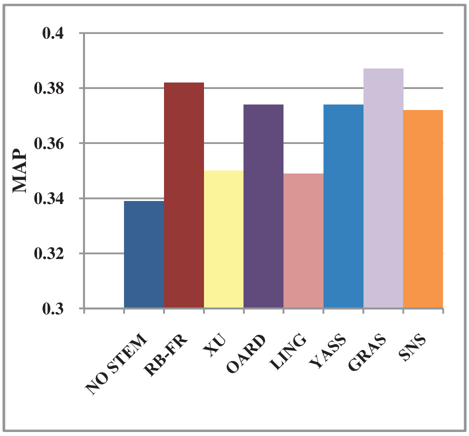

Fig. 6. MAP values for French Test Collection (CLEF 2005, 2006).

Table X. Information Retrieval Results for Bengali (FIRE 2010 Collection)

METHOD	NO STEM	RB-BE	XU	LING	YASS	GRAS	SNS	HPS
MAP	0.361	0.410 (13.3)	0.407 (12.7)	0.386 (6.9)	0.425 (17.7)	0.437 (21.1)	0.430 (19.1)	0.428 (18.6)
RELEVANT DOCUMENTS	2173	2264 (4.2)	2279 (4.9)	2269 (4.4)	2272 (4.6)	2272 (4.6)	2275 (4.7)	2272 (4.6)

Table XI. Information Retrieval Results for Marathi (FIRE 2008 Collection)

METHOD	NO STEM	RB-MA	XU	LING	YASS	GRAS	SNS	HPS
MAP	0.292	0.309 (5.8)	0.410 (40.4)	0.374 (28.2)	0.351 (20.2)	0.423 (44.9)	0.438 (50.0)	0.360 (23.6)
RELEVANT DOCUMENTS	1463	1472 (0.6)	1621 (10.8)	1557 (6.4)	1546 (5.7)	1670 (14.1)	1653 (13.0)	1561 (6.7)

stemmers performed equally well and improved MAP by nearly 13% as compared to no stemming. LING, however, showed a small improvement of 6.9% in MAP.

Marathi, being as morphologically rich a language as Bulgarian, reported large improvements of nearly 50% in MAP as compared to no stemming. SNS showed maximum improvement of 50% in MAP followed by GRAS, which improved MAP by 44.9%. XU also performed significantly well and improved MAP by 40.4%. The YASS and HPS stemmers did not perform as well in Marathi as in other languages and reported improvements of 20.2% and 23.6%, respectively, in MAP. Interestingly, the Rule-Based Marathi stemmer performed the worst among all the stemmers. Figures 7 and 8 show a comparison of MAP values of various stemmers for Bengali and Marathi, respectively.

6.4. Analysis of Results

In this subsection, we present our findings on the basis of analysis of experimental results. Among rule-based stemmers for English, the LOVIN stemmer did not perform well as compared to other statistical stemmers because it uses an incomplete list of suffixes and cannot handle all the variant word forms in the corpus. The performance of the PORTER stemmer is found to be average in Information Retrieval tasks. The reason for the decrease in performance in some cases is that while removing suffixes

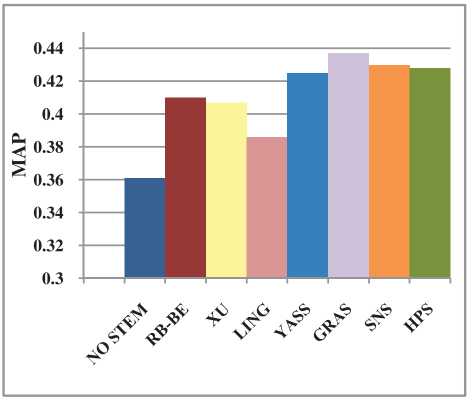

Fig. 7. MAP values for Bengali Test Collection (FIRE 2010).

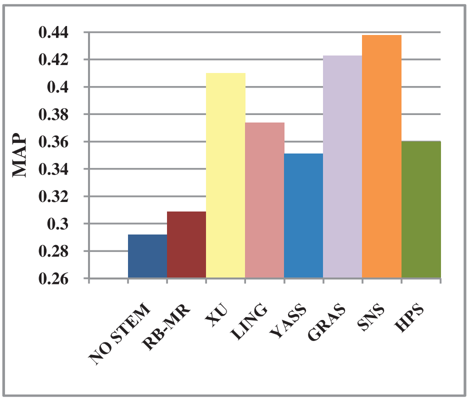

Fig. 8. MAP values for Marathi Test Collection (FIRE 2008).

the Porter stemmer removes certain valid word endings by considering them suffixes and is unable to conflate similar words in certain cases, whereas these problems are properly handled through context or co-occurrence information of the corpus. Various rule-based stemmers for European and Asian languages performed inferior to most of the statistical stemmers probably because they cannot handle all the inflections caused by verb forms and missed many commonly occurring derivational suffixes.

Among lexicon analysis-based stemmers, GRAS gave the best results of all the languages, but it lags behind a few corpus analysis-based stemmers (SNS and QBS). The GRAS stemmer is highly concerned with improving the recall rate and tends to over-stem the words. The two suffix frequency-based stemmers, FCB and OARD, gave a moderate performance in retrieval experiments. The reason for their average performance is that, while identifying the suffixes from the corpus, they miss many infrequent suffixes (suffixes whose frequency is below cutoff). The string similarity-based word clustering method, YASS, performed significantly well in all the languages except Marathi. This is due to the fact that the complete linkage clustering method used in YASS cannot handle large equivalence classes and divides large classes into a number of clusters. But equivalence classes in the Marathi language are large due to the presence of long suffixes. The Minimum Description Length-based morphological analyzer, LING, is found to be computationally most intensive as compared to other stemmers of the same category.

The corpus analysis-based stemmers are found to be quite effective in retrieval tasks, as they handle many under-stemming errors using corpus-specific information. Among the two co-occurrence statistic-based stemmers, XU and SNS, SNS performed better in all the languages. XU did not perform well in morphologically complex languages because the optimal partitioning algorithm is not feasible for languages with large-size equivalence classes. The Query-Based stemmer (QBS) performed the best among all the stemmers of this category. This is probably due to the fact that Query-Based stemmer controls the over-stemming errors by reducing the influence of the variants that are not related to the original query context. The High-Precision stemmer (HPS) performed significantly well and showed a high out-of-vocabulary rate.

7. COMPARISON OF STEMMERS IN LANGUAGE MODELING AND NLP APPLICATIONS

The performance of stemmers is usually evaluated through Information Retrieval tasks. But a versatile stemmer should perform well in a variety of tasks. The success

Table XII. Impact of Incorporating Different Stemmers in Trigram Language Modeling

METHOD	**NO STEM**	**LOVIN**	**PORTER**	**YASS**	**GRAS**	**SNS**	**HPS**
PERPLEXITY	271	219	217	229	227	240	238

and failure rates in different tasks uncover the characteristics and degree of versatility of various stemming algorithms. For instance, if a stemming algorithm performs well in Information Retrieval tasks but relatively poorly in inflection removal tasks, then it implies that the stems produced by the stemming method do not correspond to the lemmas. So this information may suggest the nature of tasks in which the stemming method will be useful. In this section, we intend to study the impact of stemmers in three different tasks, namely Language Modeling, POS Tagging, and Text Classification. On the basis of IR results presented in Section 6.3, we chose two best-performing stemmers each from the language-specific, lexicon analysis, and corpus analysis categories. Specifically, we considered the following stemmers in our experiments.

(1) Language Specific Stemmers: Porter Stemmer [Porter 1980] and Lovins Stemmer [Lovin 1968].
(2) Lexicon Analysis-Based Stemmers: GRAS [Paik et al. 2011a] and YASS [Majumder et al. 2007].
(3) Corpus Analysis-Based Stemmers: Co-occurrence SNS [Paik et al. 2011b] and HPS [Brychcin and Konopik 2015].

7.1. Language Modeling

Statistical language models calculate the probability of a sequence of words or sentences. Ways to estimate relative probabilities of different words in the sentence is extremely useful in many Natural Language Processing applications. The performance of various applications such as Machine Translation, Spelling Correction, Handwriting Recognition, Question Answering, Speech Recognition, Summarization, and so on, greatly relies on the language model being used. Any improvement in the language model can further improve the performance of the task in which the language model is being used. A number of studies [Allan and Kumaran 2003; Brychcín and Konopík 2011] have shown that the morphological knowledge such as lemmatization, stemming, and POS tagging notably improves the perplexity of the language model. Stemming in language modeling is used as a type of smoothing of probability estimation. Stemming increases the count of word occurrences, which is used to approximate the likelihood of a word. Recently, Brychcín and Konopík [2015] verified that stemming helps in improving language modeling by making use of different stemmers.

In our language modeling experiments, we first estimated the baseline model using unstemmed corpus and then estimated different language models on the text stemmed using various language-specific and statistical stemmers used in our analysis. All the estimated models are n-gram language models that find the conditional probability of the word from its history (i.e., n-1 previous words). The value of n in our experiments is taken as 3, and we have used modified Kneser-Ney smoothing [Chen and Goodman 1998]. The training and testing text corpus are taken from the TREC Wall Street Journal Corpus, and the same collection is used for the baseline and other language models. The effect of different stemmers on the perplexities of language models is presented in Table XII. Perplexity can be viewed as the confusion of the language model. One language model is better than the others if it gives lower perplexity to the test collection. The low value of perplexity implies superior prediction capability of the model. A number of studies in different applications such as speech recognition [Watanabe et al. 2011] and machine translation [Brychcín and Konopík 2014] have

Table XIII. Effect of Various Stemmers on Tagging Accuracy

Method	Maximum Entropy-Based Tagger	Perceptron-Based Tagger
NO STEM	95.46%	96.65%
LOVIN	95.01% (−0.45)	96.27% (−0.38)
PORTER	**96.67% (+1.21)**	**97.78% (+1.13)**
YASS	95.02% (−0.44)	96.60% (−0.05)
GRAS	95.78% (+0.32)	96.98% (+0.33)
SNS	95.97% (+0.51)	97.15% (+0.50)
HPS	96.39% (+0.93)	97.32% (+0.67)

*The values in brackets denote change in accuracy as compared to no stemming.

shown that a decrease in perplexity helps in improving the performance of the entire task.

We can see from the Table XII that all the stemming methods reduced the perplexity of the language model. The rule-based stemmers, PORTER and LOVIN, performed better than statistical stemmers, as both stemmers reduced the perplexity of the model by more than 50. Among lexicon analysis-based stemmers (YASS and GRAS) performed better than corpus analysis-based stemmers (SNS and HPS) by assigning lower perplexity value to the same test corpus.

7.2. Part-of-Speech Tagging

POS Tagging is the process of labeling the words of the corpus with their corresponding syntactic category depending on their definition and context. The syntactically disambiguated data through POS tagging is useful in many applications like Information Extraction, Speech Processing, Phrase Chunking, Word Sense Disambiguation, and Bioinformatics, and so on. Morphological knowledge such as Lemmatization and Stemming is found to be advantageous in the design of POS taggers as the mapping of inflected forms to their base form during the training phase enhances the probability of finding the correct tag for the word [Shrivastava and Bhattacharyya 2008].

In order to analyze the impact of stemming in POS tagging, we used Maximum Entropy [Ratnaparakhi 1996] and Perceptron-based machine-learning models. These models estimate the correct part-of-speech tag out of the tag set using a probability model. The experiments are performed on the Penn Treebank Wall Street Journal Corpus [Marcus et al. 1994] using the opennlp toolkit (available at http://opennlp.apache.org/). In our experiments, we first stemmed the corpus using different stemmers and then trained the tagger using the MaxEnt and Perceptron models. The effect of various stemmers on the tagging performance is compared in Table XIII. The results are expressed in terms of tagging accuracy, that is, the number of tags correctly labeled by the tagger.

Table XIII shows that stemming improves the tagging accuracies of both taggers. As is clear from the table, the PORTER stemmer reported maximum improvement of 1.21% and 1.13% in the accuracy of MaxEnt-based and Perceptron-based taggers, respectively. But another rule-based stemmer under analysis, that is, the LOVIN stemmer, did not perform well and decreased the accuracy by nearly 0.4%. Among statistical stemmers, corpus analysis-based stemmers (HPS and SNS) performed better than lexicon analysis-based stemmers (GRAS and YASS). HPS showed an improvement of 0.93% using the MaxEnt tagger and +0.67% using the Perceptron tagger. SNS and GRAS reported an improvement of 0.5% and 0.3%, respectively, in the case of both of the taggers. YASS, on the other hand, did not perform well in both cases and decreased the tagging accuracy.

Table XIV. Classification Results for 20NG and NYTimes Using the MaxEnt Classifier

	New York Times (NYTimes)		**20 Newsgroups (20NG)**	
Method	**Precision**	**Recall**	**Precision**	**Recall**
NO STEM	0.4096	0.3948	0.3988	0.2969
LOVIN	0.3681 (−10.1)	0.3407 (−13.7)	0.3787 (−5.1)	0.2876(−3.1)
PORTER	0.4585 (11.9)	0.3992 (1.1)	0.4108 (3.0)	0.3028 (2.0)
YASS	0.4515 (10.2)	0.4074 (3.2)	**0.4263 (6.9)**	**0.3240 (9.1)**
GRAS	**0.4737 (15.6)**	**0.4189 (6.1)**	0.4227 (6.0)	0.3150 (6.1)
SNS	0.4415 (7.8)	0.4042 (2.4)	0.4163 (4.3)	0.3123 (5.1)
HPS	0.4427 (8.1)	0.4007 (1.5)	0.4187 (5.0)	0.3150 (6.1)

*The values in brackets denote relative change as compared to no stemming.

7.3. Text Classification

Automatic text classification of documents into predefined categories has been an important application in managing a large amount of documents that are available in digital form and are increasing continuously. Text classification methods have been applied to various applications such as email routing, spam detection, readability assessment, author identification, language identification, genre classification, and so on. Most of the automatic text classification methods are based on machine-learning techniques, which automatically build a classifier using pre-classified text. Initial studies [Riloff 1995; Baker and McCallum 1998] have reported that stemming does not help in increasing classification accuracy, but subsequent studies [Cohen et al. 2005; Bhamidipati and Pal 2007] found stemming advantageous. So recent tendency is to apply stemming at the pre-processing stage of text classification, as it reduces the stochastic dependence between terms and the dimensionality of the feature set [Sebastiani 2002].

In order to analyze the effect of stemming on text classification accuracy, we used a Maximum Entropy– (MaxEnt) [Nigam et al. 1999] based classifier. A maximum entropy classifier is a probabilistic classifier that finds the conditional distribution of class labels given in a document. These classifiers do not assume conditional independence of various features. All the experiments are performed using the package RTextTools [Jurka et al. 2013] on two datasets. The first dataset is the 20 Newsgroups (20NG) dataset, which is a collection of 19,997 news documents divided evenly into 20 different newsgroups. The 20 NG data collection is commonly used in machine-learning tasks, particularly text classification and text clustering. The second dataset used in the experiments is the NYTimes dataset provided in the R package. The dataset contains labeled front-page headlines of *The New York Times* articles compiled by Professor Amber E. Boydstunat at the University of California at Davis. The headlines are manually classified into 27 different categories. Both the datasets are divided randomly into two parts for the training and testing phases, and the same division is used for all the stemmers. Sixty percent of the documents are used in the training phase and 40% are used in the testing phase. The classification accuracies for both datasets are presented in Table XIV in terms of Precision and Recall. Precision refers to the percentage of classified documents that are correct, whereas Recall refers to the proportion of documents in a class that the classifier assigns to that class.

It is clear from Table XIV that, except for the LOVIN stemmer, all other stemmers improved the classification results of the MaxEnt classifier on both datasets. The lexicon analysis-based stemmers, YASS and GRAS, performed better than the rule-based and corpus analysis-based stemmers on both datasets. In the case of the 20NG dataset, YASS showed a maximum increase of 6.9% in precision and 9.1% in recall, followed by GRAS, which reported an increase of 6% in both precision and recall. The orpus analysis-based stemmers, SNS and HPS, performed almost equally, whereas the PORTER stemmer performed slightly worse compared to statistical stemmers.

In the NYTimes dataset, there is a greater improvement in classification accuracy compared to the 20NG dataset. GRAS reported a maximum increase of 15.6% in precision and 6.1% in recall. PORTER and YASS performed consistently well and showed an increase of 11.9% and 10.2%, respectively, in precision. The corpus analysis-based stemmers, SNS and HPS, performed worse than lexical analysis-based stemmers and reported an increase of nearly 8% in precision.

8. APPLICATIONS OF TEXT STEMMING

Text Stemming is widely used as a part of the text pre-processing step in Information Retrieval and Natural Language Processing systems. Stemming is employed in text pre-processing stage to solve the problem of vocabulary mismatch and reduction in the dimensionality of representation set or training data. In this section, we discuss some systems where text stemming is employed at the pre-processing stage.

(1) *Document Classification and Clustering.* Document classification systems label the natural language texts with pre-defined categories or classes and are widely used in applications like automatic indexing, text filtering, categorization of web pages, and so on. Document clustering systems group similar documents into clusters but are not labeled with pre-defined classes as in the case of classification systems. In text classification and clustering systems, the documents are often represented as a Vector Space Model. So conflating similar words is always preferred, as it reduces the dimensionality of the feature set. A number of studies [Gaustad et al. 2002; Rosell 2003; Bhamidipati and Pal 2007; Froud et al. 2010] have shown that reducing the morphological variants to their stems in the documents significantly improves the classification and clustering accuracy.

(2) *Automatic Summarization Systems.* Automatic Summarizers produce condensed representation of the input text while preserving the important information content. Stemming is found to be an important part of the text pre-processing step of these systems as it reduces the dimensionality of the representation set. Louis and Nenkova [2009] studied the effect of stemming on the performance of summarization systems and verified that their performance improves significantly when stemming is done at the pre-processing stage.

(3) *Machine Translation Systems.* Machine Translation is a subfield of computational linguistics that uses computer software to convert text or speech in one natural language into another. The performance and computational complexity of Machine Translation systems are improved by stemming the variant word forms. Stemming reduces the sparsity in the language model and the translation tables, thereby improving the performance of the machine translation system by correctly recovering the sequence of stems in the target language [Toutanova et al. 2008]. Goldwater and McClosky [2005] verified that in statistical machine translation systems, stemming is helpful in the appropriate estimation of word-to-word alignment probabilities. Stemming also helps in reducing the out-of-vocabulary rate in the test set [Bisazza and Federico 2009] and minimizing the lexical granularities of source and target language, which can produce a better model of the translation task.

(4) *Question Answering Systems.* Question answering systems aim to find out the exact answers to the natural language questions in a large document collection. The issue of vocabulary mismatch between the questions and the answers is resolved through stemming as it figures out different morphological variant forms. So the use of a stemmer is of high importance in these systems. Several studies have suggested

that indexing and searching the collection using stems significantly increases the performance of these systems in terms of both precision and recall [Hammo et al. 2002; Monz 2003; Akour et al. 2011].

(5) *Part-of-Speech Tagging Systems.* Part-of-Speech tagging systems are programs that label each word of the input text to its most probable part of speech according to the context. These systems are widely used in various Natural Language Processing applications. In morphological analysis such as lemmatization, stemming has been extensively used in developing POS tagging systems. Stemming increases the probability of finding the correct syntactic category of the word by mapping the variant word forms to its base form. Several Part-of-Tagging systems have been developed that make use of stemming to harness the morphological richness of the language and do not employ any other tools like structured lexicon or morphological analyzers [Khoja 2001; Al Shamsi and Guessoum 2006; Shrivastava and Bhattacharyya 2008].

(6) *Word Sense Disambiguation.* Word sense disambiguation is the ability to determine the meaning or sense of the word according to its context in the sentence. The morphological variant word forms are not only relationships between the words but also between the word senses [Krovetz 1993]. Stemmer can use these relationships to resolve the meaning of the word. For instance, stemmer can identify the correct meaning of the word in the sentence using the part-of-speech or irregular morphology knowledge.

(7) *Text Searching.* In searching systems, the user must correctly predict the word form to be used in the query to get the best results. With stemming, the searcher need not worry about the exact word form. Stemming helps in improving the performance of text searching systems by performing context-sensitive analysis on the query and document sides. Context-sensitive stemming [Jenkins and Smith 2005; Peng et al. 2007] on the query side helps in predicting the word forms that are better than the original query terms and expanding the terms with only those forms. Similarly, on the document side, context-sensitive document matching is performed that matches the documents according to the expanded terms in the query [Peng et al. 2007]. Context-sensitive stemming not only improves precision by stemming terms according to the context of the query term but also reduces computational cost and query traffic.

Besides the systems mentioned above, text stemming is also employed in many other Natural Language Processing applications such as entity identification [Zitouni et al. 2005; Konkol and Konopik 2014], spelling checkers [Islam et al. 2007], keyword extraction [Hulth 2003], and so on.

9. MORPHO CHALLENGE COMPETITIONS

The learning of morphology helps in the morphological segmentation of words, grouping of morphologically related words, and complete morphological analysis of word forms. These tasks are very useful in Natural Language and Information Retrieval applications. A number of studies [Virpioja et al. 2011; Hammarstrom and Borin 2011] related to supervised, unsupervised, or semi-supervised learning of morphology of language have been reported in the literature. Research into unsupervised methods of morphological analysis has been improved with the organization of annual morpheme extraction competitions. These competitions aim to design methods to discover the various morphemes of word forms. In this section, we describe various Morpho Challenge Competitions organized by the Computer Science Department, Aalto University, and the Forum of Information Retrieval of India (FIRE) for European and Asian languages.

9.1. Review of Morpho Challenge Competitions

The first Morpho Challenge Competition was in 2005, where the objective was to obtain unsupervised segments of the word forms in three languages, namely English, Turkish, and Finnish. These segments were evaluated in two ways. First, the segments submitted by the competitors were compared against gold standards. Second, the segments were used to train the *n*-gram language model, which is used in automatic speech recognition. In 2007, the task was to provide full morphological analysis of the various word forms. German was also included along with three languages of the 2005 Morpho Challenge. The morpheme analysis was evaluated by comparing against gold standards and in IR experiments by replacing the words with the morpheme analysis submitted by the competitors. In 2008, the task and evaluation metrics were same as that of 2007. The Arabic language was also included this year. In 2009, the task was set to return complete morphological analysis of words. But this year, the morphological analysis was also evaluated by using them in two Statistical Machine Translation applications: Finnish to English and German to English. The words in the source language (German and Finnish) were replaced with morpheme analysis, whereas the words of the target language English were not changed. In 2010, the tasks and evaluation methods were same as that of 2009 and the experiments were performed in English, Finnish, Turkish, and German. The complete report of datasets, evaluation results, and the list of participants can be reviewed from the annual evaluation reports of the challenges [Kurimo et al. 2006, 2008, 2009a, 2009b, 2010].

The Forum for Information Retrieval Evaluation of India (FIRE) also organized a number of Morpheme Extraction Tasks (2012–2014) to promote the evaluation of various morpheme extraction techniques on Indian languages. The first morpheme extraction task by FIRE was organized in 2012, wherein the task was set to obtain morphemes from the list of word forms in five languages, namely Hindi, Bengali, Marathi, Odia, and Gujarati. The morphemes were evaluated in Information Retrieval tasks. In 2013, Tamil was also included, and linguistic evaluation against gold standards was done for Tamil and Bengali along with evaluation in IR experiments. In 2014, a morpheme extraction task was evaluated using linguistic evaluation for three languages Bengali, Gujarati, and Tamil. The complete report of datasets, evaluation results, and list of participants can be reviewed from the annual reports of the competitions [Sankepally 2013; Gandhi et al. 2014]. Table XV summarizes the evaluation tasks and languages selected for various Morpho Challenge competitions.

9.2. Evaluated Algorithms and Results

In Morpho Challenge competitions 2005–2010, more than 50 morpheme analysis algorithms were proposed and evaluated. We here describe the algorithms that gave the best evaluation results.

- —*Bernhard Method*. This method is proposed by Bernhard [2006] and has been used in Morpho Challenges 2005 and 2007. It has been reported to be one of the best methods in various linguistic and application-oriented evaluations for English, Finnish, and Turkish. It first finds an initial set of prefixes and suffixes using count of substrings and then finds a set of possible segments of the word forms. The best segment is then chosen using a cost function based on the A* algorithm. The author submitted two methods (*Bernhard 1 and Bernhard 2*) by making use of two different cost functions.
- —*Morfessor Baseline*. This is the baseline algorithm [Creutz and Lagus 2002] that employs an MDL-based cost function to minimize the size of the morph vocabulary and the words in the corpus. The method performed significantly well in linguistic evaluations and other applications. In Automatic Speech Recognition, it showed the smallest letter error rate.

Table XV. Summary of Morpho Challenge Competitions

Year	Languages	Task	Evaluation
2005	English, Finnish, Turkish	Unsupervised Segmentation	Comparison with gold standard; Automatic Speech Recognition
2007	English, French, German, Turkish	Full Morphological Analysis	Comparison with Gold Standard, Information Retrieval experiments.
2008	English, French, German, Turkish, Arabic	Full Morphological Analysis	Comparison with Gold Standard, Information Retrieval experiments.
2009	English, French, German, Turkish, Arabic	Full Morphological Analysis	Comparison with Gold Standard, Information Retrieval, and statistical machine translation experiments
2010	English, French, German, Turkish	Full Morphological Analysis	Comparison with Gold Standard, Information Retrieval, and statistical machine translation experiments
2012	Hindi, Bengali, Marathi, Gujarati, Odia	Morpheme Segmentation	Information Retrieval experiments.
2013	Hindi, Bengali, Marathi, Gujarati, Odia, Tamil	Morpheme Segmentation	Comparison with Gold Standard, Information Retrieval experiments.
2014	Bengali, Gujarati, Tamil	Morpheme Segmentation	Comparison with Gold standard,

—*Morfessor Categories-MAP*. This is an extension of the Morfessor baseline algorithm proposed by Creutz and Lagus [2005]. It uses HMMs to represent various morph categories and then a maximum *a posteriori* (MAP) framework is utilized to optimize these representations. The proposed method reported a slight improvement in linguistic evaluations for almost all the languages but did not perform well in Information Retrieval and Machine Translation applications.

—*Allomorfessor*. Allomorfessor [Virpioja et al. 2009; Kohonen et al. 2008] is also an extension of the Morfessor baseline with a MAP model and was used in Morpho Challenges 2008 and 2009. It is based on the principle that surface forms of various morphemes are usually alike, and the variation occurs near the boundary. So, it models allomorphemes by mutations. The method did not show any improvement in Information Retrieval and Machine Translation tasks but reported improvements in linguistics evaluations.

—*ParaMor*. Monson et al. [2007] proposed this morpheme induction method that uses unsupervised framework for inflectional variations of the words. It has been combined with Morfessor using various combination methods. *ParaMor-Morfessor* is a combination of the Morfessor Categories MAP approach and the Paramor approach that has been used in Morpho challenges 2007 and 2008. *Paramor-Morfessor Union* is a combination of the Morfessor approach and the Paramor approach by considering the union of the various segment points. *ParaMor-Morfessor Mimic* and *Paramor Mimic* are average probabilistic models.

—*RePortS*. Keshava and Pitler [2006] proposed a morpheme induction method that uses a letter successor variety–based method that builds forward and backward trees. These trees are used to score the prefixes and suffixes to get a list of candidate morphemes. The set of candidate morphemes are pruned, and segments of the test words are determined.

—*Promodes*. Speigler et al. [2009] proposed an unsupervised morpheme extraction algorithm that uses a probabilistic generative model. It views the boundaries of the morphemes as hidden variables and incorporates letter successor varieties estimates in the segments. The proposed method did not perform well in Information Retrieval and Machine Translation tasks, but in Arabic linguistic evaluation it provided the best results.

—*Zeman*. An unsupervised morpheme segmentation method was proposed by Zeman [2007] that identifies frequently occurring stem and derivations from the corpus assuming that each word has one stem and one suffix. The method has further been enhanced in Zeman [2008], in which prefixes are also identified using two different techniques. This method achieved the best results in the Morpho Challenge 2007 for the Turkish language.

—*DEAP*. Spiegler et al. [2010] proposed a deductive-abductive parsing method of unsupervised morpheme segmentation. It makes use of context-free grammar for transforming the morphological analysis into rules. The categories of various morphemes are hypothesized, and the best hypothesis is chosen using MDL or the probability approach. Two different variations are used, No CATegory (NOCAT) and CATegory (CAT). In the case of NOCAT, the morpheme labels are deleted, and in the case of CAT, these labels are considered.

—*Morfessor Semi-Supervised Versions*. A number of semi-supervised alternates for Morfessor baseline have been proposed [Kohonen et al. 2010] in the literature. *Morfessor U+W* learns the weight of data estimates from the baseline model. *Morfessor S+W* learns the noted segments for the searching method. *Morfessor S+W+L* use already annotated data for labeling segments.

In Morpheme extraction tasks organized by FIRE for three consecutive years, nearly 17 supervised, unsupervised and semi-supervised systems have been proposed and evaluated on standard FIRE collections for various Indian languages. The methods that provided good results are described below.

—*YASS*. Yet Another Suffix Stripper is an unsupervised stemming method (described in Section 5.1.3) proposed by Majumder et al. [2007]. It clusters various morphologically related words using string distances that support the longest-match principle and avoid early matching.

—*ISM*. This is another unsupervised stemming system proposed by Yadav et al. [2012] that identifies potential suffixes from the corpus using their frequency of occurrence.

—*Tamil Morphological Analyzer (TMA)*. This system was developed by Devi et al. [2013] and finds root and other inflectional knowledge of words using a paradigm-based method. The system has been developed as a finite-state machine.

—*Rule-Based Stemmers for Bengali and Hindi*. Ganguly et al. [2012] developed a supervised analyzer for Bengali and Hindi that deletes suffixes using grammatical inflection rules for both the languages. The system performed well in Information Retrieval experiments for Hindi.

—*IndiLem*. Chakrabarty et al. [2014] proposed an unsupervised morphological analysis system that first generates a trie network and then discovers lemmas by searching the structure. The system gave promising results in linguistic evaluation for Bengali.

—*AMRITA*. Kumar et al. [2014] proposed machine-learning-based morphological analyzer. It analyzes various word forms from a POS tagged input using a suffix-based and supervised machine-learning approach.

Table XVI presents the results of the linguistic evaluations of the various morpheme extraction competitions organized for the European and Asian languages. For each

Table XVI. Results of Linguistic Evaluations of Various Morpho Challenge Competitions (Results of Best Two Submitted Methods Are Reported in Each Language)

	Language	Method	Precision	Recall	F-Score
Morpho Challenge 2005	English	ReportS	76.2	77.4	76.8
	English	Bernhard 1	67.7	65.5	66.6
	Finnish	Bernhard 2	63.0	66.4	64.7
	Finnish	Bernhard 1	73.6	55.6	63.3
	Turkish	Bernhard 2	65.4	65.2	65.3
	Turkish	Bernhard 1	77.9	42.8	55.9
Morpho Challenge 2007	English	Bernhard 2	61.6	60.0	60.8
	English	Bernhard 1	72.1	52.5	60.7
	Finnish	Bernhard 2	59.6	40.4	48.2
	Finnish	Morfessor MAP	76.8	27.5	40.5
	Turkish	Morfessor MAP	76.4	24.5	37.1
	Turkish	Zeman	65.8	18.8	29.2
	German	Paramor + Morfessor	51.4	55.5	53.4
	German	Bernhard 2	49.1	57.4	52.9
Morpho Challenge 2008	English	Paramor + Morfessor	50.7	63.3	56.2
	English	Morfessor Baseline	71.9	43.3	54.0
	Finnish	Paramor + Morfessor	49.8	47.2	48.5
	Finnish	Morfessor MAP	76.8	27.5	40.5
	Turkish	Paramor + Morfessor	51.9	52.1	51.9
	Turkish	Paramor	56.7	39.4	46.5
	German	Paramor + Morfessor	49.5	59.5	54.1
	German	Morfessor MAP	67.6	36.9	47.7
	Arabic	Paramor + Morfessor	79.8	27.4	40.8
	Arabic	Morfessor Baseline	78.1	23.7	36.4
Morpho Challenge 2009	English	Allomorfessor	68.9	56.8	62.3
	English	Morfessor Baseline	74.9	49.8	59.8
	Finnish	ParaMor-Morfessor Union	47.9	50.9	49.3
	Finnish	ParaMor-Morfessor Mimic	51.8	45.4	48.3
	Turkish	ParaMor-Morfessor Mimic	48.1	60.3	53.5
	Turkish	ParaMor-Morfessor Union	47.2	60.0	52.8
	German	ParaMor-Morfessor Union	52.5	60.2	56.1
	German	ParaMor-Morfessor Mimic	51.0	57.7	54.2
	Arabic	Letters	70.4	53.5	60.8
	Arabic	PROMODES	76.9	37.0	50.0
Morpho Challenge 2010	English	Morfessor S+W	65.6	69.3	67.4
	English	Morfessor S+W+L	67.9	69.2	67.1
	Finnish	DEAP MDL-NOCAT	56.0	70.7	62.5
	Finnish	DEAP MDL-CAT	57.4	66.6	61.6
	Turkish	Morfessor S+W+L	71.7	59.9	65.3
	Turkish	DEAP MDL-NOCAT	68.9	56.1	61.8
	German	Morfessor U+W	58.5	44.9	50.8
	German	Morfessor CatMAP	72.7	35.4	47.6
FIRE MET 2013	Tamil	TMA	84.3	88.1	86.1
	Tamil	ISM	80.2	18.9	30.5
	Bengali	ISM	60.6	32.1	42.1
MET 2014	Tamil	AMRITA	10.9	40.9	17.1
	Bengali	IndiLem	56.1	65.1	60.3

language, the results of the two best submitted methods are reported. In some cases (Tamil and Bengali), only one result is shown because of a single competitor.

10. CHALLENGES IN UNSUPERVISED STATISTICAL STEMMING

There have been a lot of conceptual advancements in unsupervised stemming methods. As already discussed, statistical stemming techniques obviate the need for any linguistic resource, as they group morphologically related words from the ambient corpus. So they are good substitutes to rule-based stemmers, especially for resource-scarce languages. Despite all these benefits, unsupervised statistical stemming techniques have certain open issues and challenges that deserve attention from the researchers in the area. In this section, we describe some open issues related to unsupervised stemmers.

One major weakness of the existing statistical techniques is that they can handle only a class of suffixing languages. These stemmers only consider the inflectional variations due to the addition of suffixes. But there are a number of other linguistic processes, such as compounding, conversion of words, and so on, that change significant parts of the words. The rules that cause these inflectional variations occur quite frequently in natural language text collections. So the future research must focus on discovering ways to identify the variants that are formed by modifying significant parts of the words and not just suffixes.

Another issue is related to the size and nature of the corpus employed in the development of the stemmer. Corpus-based stemmers involve a lot of computations and complex data-mining methods. An appropriate corpus size would reduce computations and time involved in the development of these stemmers. So researchers in this field must evaluate the performance of their stemmers with different corpus size by taking sub-samples from the corpus to decrease the excessive computational load. Recently, Brychcin and Konopík [2015] evaluated the performance of their stemmer with a training data ranging from 50,000 words to 15,000,000 words and reported that good results could be achieved with only 50,000 words. Moreover, the nature of the corpus employed also affects the performance of the final stemmer. A corpus that contains documents from different areas like engineering, medicine, religion, fiction, and so on, In addition to only news documents can enrich the vocabulary of the corpus. Hence, the study of the nature and the size of the corpus while developing unsupervised stemmers is an important concern that needs to be addressed.

Another major issue related to unsupervised stemming is to discover ways to identify advanced semantic relationships in the corpus to group semantically related words. Most of the current statistical stemming methods strip known affixes from the word forms in each case. But it is necessary to determine the cases in which the affix can or cannot be removed from the word. For instance, stripping off the suffix “ing” in the case of word pairs *listen* and *listening* is a correct stemming action but the removal of “ing” in the case of *shin* and *shining* would be a stemming error. So methods that model advanced semantic relations need to be investigated to further improve the performance of statistical stemmers.

Another major limitation of corpus-based stemmers is that these stemmers are quite sensitive to parameters. The various techniques used for the development of these stemmers require parameter or threshold tuning that is dependent on a number of factors, such as the nature of the language, the corpus type of association measure used, and so on. For example, the hierarchal clustering algorithm used for grouping morphological variant words in stemming algorithms depends on a similarity threshold to decide the number of clusters. Therefore, researchers must explore techniques that are quite insensitive to parameters to fully automate the stemming process.

Another major issue is related to the evaluation of stemmers individually and independently of any specific application. The performance of existing stemmers is usually evaluated indirectly by using them as a pre-processor in Information Retrieval systems. But in IR systems, there are many components, and the complex interaction between the different components of the retrieval system makes it difficult to evaluate stemming algorithms. Some direct evaluation metrics have also been proposed to measure the performance of stemmers without the necessity to embed them in an Information Retrieval system such as a reduction in the input vocabulary size or classifying the various types of errors that are generated. All these direct evaluation metrics measure the performance in a relative manner and do not provide an absolute and objective metric to measure stemmer's strength.

11. CONCLUSION

Text Stemming is a long-studied technology with a wide range of applications. In this article, we presented an exhaustive review of stemmers over the past 50 years, with an important look at the issues for stemming in non-English languages. We analyzed the performance of some well-known rule-based and corpus-based stemmers in Information Retrieval tasks using standard test collections and highlighted their merits and demerits. Besides Information Retrieval, the performance of stemmers has also been analyzed in Language Modeling and Natural Language Processing applications. The analysis of stemming methods in different scenarios will help the users to choose a stemmer according to the type of application. Moreover, the challenges and open problems related to unsupervised stemming outlined at the end of the article will give future directions to the researchers to enhance the performance of unsupervised corpus-based stemming methods.

ACKNOWLEDGMENTS

We thank all the anonymous reviewers for their valuable comments and suggestions for improving the article.

REFERENCES

Giorgos Adam, Konstantinos Asimakis, Christos Bouras, and Vassilis Poulopoulos. 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*. 389–397.

Farag Ahmed and Andreas Nürnberger. 2009. Evaluation of n-gram conflation approaches for Arabic text retrieval. *J. Am. Soc. Inf. Sci. Technol.* 60, 7, 1448–1465.

Mohammed Akour, Sameer Abufardeh, Kenneth Magel, and Qasem Al-Radaideh. 2011. QArabPro: A rule based question answering system for reading comprehension tests in Arabic. *Am. J. Appl Sci.* 8, 6, 652.

Qurat-ul-Ain Akram, Asma Naseer, and Sarmad Hussain. 2009. Assas-Band, an affix-exception-list based Urdu stemmer. In *Proceedings of the 7th Workshop on Asian Language Resources*. 40–46.

Fatma Al Shamsi and Ahmed Guessoum. 2006. A hidden Markov model-based POS tagger for Arabic. In *Proceeding of the 8th International Conference on the Statistical Analysis of Textual Data, France*. 31–42.

Mohammed Al-Kabi. 2013. Towards improving khoja rule-based arabic stemmer. In *Proceedings of the IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)*. 1–6.

James Allan and Giridhar Kumaran. 2003. Details on stemming in the language modeling framework. *UMass Amherst CIIR Tech. Report–IR-289.*

Reinaldo Alvares, Ana Garcia, and Inhauma Ferraz. 2005. STEMBR: A stemming algorithm for the Brazilian Portuguese language. In *Proceedings of 12th Portuguese Conference on Artificial Intelligence, EPIA 2005*. 693–701.

Gianni Amati and Cornelis J. Van Rijsbergen. 2002. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Trans. Inf. Syst*. 20, 4, 357–389.

Juhi Ameta, Nisheeth Joshi, and Iti Mathur. 2012. A lightweight stemmer for Gujarati. In *Proceedings of 46th Annual Convention of Computer Society of India.*

Sebghatullah Aslamzai and Saidah Saad. 2015. Pashto language stemming algorithm. *Asia-Pacific J. Inf. Technol. Multimed.* 4, 1.

Harald Baayen, Richard Piepenbrock, and Rijn Van. 1993. *The CELEX Lexical Data Base (CD-ROM)*, Linguistic Data Consortium.

Michela Bacchin, Nicola Ferro, and Massimo Melucci. 2002. The effectiveness of a graph-based algorithm for stemming. In *Digital Libraries: People, Knowledge, and Technology*. Springer, 117–128.

Michela Bacchin, Nicola Ferro, and Massimo Melucci. 2005. A probabilistic model for stemmer generation. *Inf. Process. Manag.* 41, 1, 121–137.

Douglas Baker and Andrew Kachites McCallum. 1998. Distributional clustering of words for text classification. In *Proceedings of 21st ACM SIGIR Conference*. Melbourne, Australia, 96–103.

Delphine Bernhard. 2006. Unsupervised morphological segmentation based on segment predictability and word segments alignment. In *Proceedings of 2nd Pascal Challenge Workshop on Unsupervised Segmentation of Words into Morphemes*. 19–24.

Narayan L. Bhamidipati and Sankar K. Pal. 2007. Stemming via distribution-based word segregation for classification and retrieval. *IEEE Trans. Syst. Man Cybernet.* 37, 2, 350–360.

Andrzej Białecki. 2004. Stempel – Algorithmic Stemmer for the Polish Language. http://getopt.org/stempel/.

Marenglen Biba and Eva Gjati. 2014. Boosting text classification through stemming of composite words. *Recent Adv. Intell. Inf.* 235, 185–194.

Arianna Bisazza and Marcello Federico. 2009. Morphological pre-processing for Turkish to English statistical machine translation. In *Proceedings of the International Workshop on Spoken Language Translation*. 129–135.

Martin Braschler and Barbel Ripplinger. 2004. How effective is stemming and decompounding for German text retrieval? *Inf. Retriev.* 7, 3–4, 291–316.

Tomáš Brychcín and Miloslav Konopík. 2011. Morphological based language models for inflectional languages. In *Proceedings of IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems*.

Tomáš Brychcín and Miloslav Konopík. 2014. Semantic spaces for improving language modeling. *Comput. Speech Lang.* 28, 1, 192–209.

Tomáš Brychcín and Miloslav Konopík. 2015. HPS: High precision stemmer. *Inf. Process. Manag.* 51, 1, 68–91.

Johan Carlberger, Hercules Dalianis, Martin Hassel, and Ola Knutsson. 2001. Improving precision in information retrieval for Swedish using stemming. In *Proceedings of 13th Nordic Conference on Computational Lnguistics (NODALIDA'01)*.

Abhisek Chakrabarty, Sharod R. Choudhury, and Utpal Garain. 2014. IndiLem@ FIRE-MET-2014: An unsupervised lemmatizer for Indian languages. In *Proceedings of Forum for Information Retrieval and Evaluation (FIRE 2014)*.

Erwin Chan. 2006. Learning probabilistic paradigms for morphology in a latent class model. In *Proceedings of the 8th Meeting of the ACL Special Interest Group on Computational Phonology and Morphology*. 69–78.

Aitao Chen and Fredric Gey. 2002. Building an arabic stemmer for information retrieval. In *Proceedings of the Text Retrieval Conference (TREC'02)*. 631–639.

Stanley F. Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Comput. Speech Lang.* 13, 4, 359–393.

Aaron Cohen, J. Yang, and William Hersh. 2005. A comparison of techniques for classification and ad-hoc retrieval of biomedical documents. In *Proceedings of the Text Retrieval Conference.*

Mathias Creutz and Krista Lagus. 2002. Unsupervised discovery of morphemes. In *Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning-Volume 6*. 21–30.

Mathias Creutz and Krista Lagus. 2005. Inducing the morphological lexicon of a natural language from unannotated text. In *Proceedings of the International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning (AKRR'05)*. 51–59.

Amitava Das and Sivaji Bandyopadhyay. 2010. Morphological stemming cluster identification for Bangla. In *Knowledge Sharing Event-I: Task 3: Morphological Analyzers and Generators*.

John L. Dawson. 1974. Suffix removal for word conflation. *Bull. Assoc. Lit. Linguist. Comput.* 2, 3, 33–46.

N. Deepamala and Parmalik R. Kumar. 2015. Kannada stemmer and its effect on Kannada documents classification. In *Proceedings of the International Conference on Computational Intelligence in Data Mining*. 75–86.

Sobha L. Devi, Marimuthu K, Vijay Sundar Ram R, Bakiyavathi T, and Amudha K. 2013. AUKBC: Tamil morphological analyser. In *Proceedings of FIRE-2013*.

Ljiljana Dolamic and Jacques Savoy. 2009a. Indexing and stemming approaches for the Czech language. *Inf. Process. Manag.* 45, 714–720.

Ljiljana Dolamic and Jacques Savoy. 2009b. Indexing and searching strategies for the Russian language. *J. Am. Soc. Inf. Sci. Technol.* 60, 12, 2540–2547.

Ljiljana Dolamic and Jacques Savoy. 2010. Comparative study of indexing and search strategies for the Hindi, Marathi, and Bengali languages. *ACM Trans. Asian Lang. Inf. Process.* 9, 3.

Samhaa El-Beltagy and Ahmed Rafea. 2011. An accuracy-enhanced light stemmer for Arabic text. *ACM Trans. Speech Lang. Process.* 7, 2, 1–22.

Osama Elrajubi. 2013. An improved Arabic light stemmer. In *Proceedings of the 3rd International Conference on Research and Innovation in Information Systems (ICRIIS'13)*. 33–38.

Somayyeh Estahbanati, Reza Javidan, and Mehdi Nikkhah. 2011. A new multi-phase algorithm for stemming in farsi language based on morphology. *Int. J. Comput. Theor. Eng.* 3, 5, 623.

Antonio Fernández, Josval Díaz, and Yoan Gutiérrez. 2011. An unsupervised method to improve Spanish stemmer. In *Natural Language Processing and Information Systems*. Springer, 221–224.

Carols Figuerola, Raquel Gomez-Diaz, Angel F. Zazo, and José-Luis Alonso-Berrocal. 2001. Stemming in spanish: A first approach to its impact on information retrieval. In *Working Notes of CLEF 2001 Workshop*. Darmstadt, Germany, 197–202.

William B. Frakes and Christopher J. Fox. 2003. Strength and similarity of affix removal stemming algorithms. *ACM SIGIR Forum* 37, 1, 26–30.

Hanane Froud, Rachid Benslimane, Abdelhamid Lachkar, and S. Alaoui Ouatik. 2010. Stemming and similarity measures for arabic documents clustering. In *Proceedings of the 5th International Symposium on Communications and Mobile Network (ISVC)*. 1–4.

Nilotpala Gandhi, Kanika Mehta, Prashasti Kapadia, Adda Roshni, Vaibhavi Sonavane, and Prasenjit Majumder. 2014. Morpheme extraction task at FIRE 2014. In *Post-Proceedings of the 6th Workshop of the Forum for Information Retrieval Evaluation*.

Debasis Ganguly, Johannes Leveling, and Gareth Jones. 2012. DCU@FIRE-2012: Rule-based stemmers for Bengali and Hindi. In *Proceedings of the 4th Workshop of the Forum for Information Retrieval Evaluation (FIRE 2012)*.

Tanja Gaustad, Gosse Bouma, and Rijksuniversiteit Groningen. 2002. Accurate stemming of dutch for text classification. *Lang. Comput.* 45, 1, 104–117.

John Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *J. Comput. Linguist.* 27, 2, 153–198.

John Goldsmith. 2006. An algorithm for the unsupervised learning of morphology. *Nat. Lang. Eng.* 12, 4, 353–371.

Sharon Goldwater and David McClosky. 2005. Improving statistical mt through morphological analysis. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*. 676–683.

Vishal Gupta and Gurpreet S. Lehal. 2011. Punjabi language stemmer for nouns and proper names. In *Proceedings of the 2nd Workshop on South and Southeast Asian Natural Language Processing (WSSANLP)*. 35–39.

Vishal Gupta. 2014. Hindi rule based stemmer for nouns. *Int. J. Adv. Res. Comput. Sci. Softw. Eng.* 4, 1, 62–65.

Margaret A. Hafer and Stephen F. Weiss. 1974. Word segmentation by letter successor varieties. *Inf. Stor. Retriev.* 10, 11, 371–385.

Harald Hammarström and Lars Borin. 2011. Unsupervised learning of morphology. *Comput. Linguist.* 37, 2, 309–350.

Bassam Hammo, Hani Abu-Salem, and Steven Lytinen. 2002. QARAB: A question answering system to support the arabic language. In *Proceedings of the ACL-02 Workshop on Computational Approaches to Semitic Languages*. 1–11.

Donna Harman. 1991. How effective is suffixing? *J. Am. Soc. Inf. Sci.* 42, 1, 7–15.

Haider Harmanani, Walid Keirouz, and Saeed Raheel. 2006. A rule-based extensible stemmer for information retrieval with application to arabic. *Int. Arab J. Inf. Technol.* 3, 3, 265–272.

Yashaswini Hegde, Shubha Kadambe, and Prashantha Naduthota. 2013. Suffix stripping algorithm for kannada information retrieval. In *Proceedings of the IEEE International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. 527–533.

Asunción Honrado, Ruben Leon, Ruairi O'Dennol, and Duncan Sinclair. 2000. A word stemming algorithm for the Spanish language. In *Proceedings of the 7th International Symposium on String Processing and Information Retrieval.* 139–145.

Rodney Huddleston. 1988. *English Grammar: An Outline*. Cambridge University Press.

David A. Hull. 1996. Stemming algorithms-A case study for detailed evaluation. *J. Am. Soc. Inf. Sci.* 47, 1, 70–84.

Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*. 216–223.

Mohammed Islam, Mohammed Uddin, and Mumit Khan. 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*.

Marie-Claire Jenkins and Dan Smith 2005. Conservative stemming for search and indexing. In *Proc. ACM SIGIR.*

Chris Jordan, John Healy, and Vlado Keselj. 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*. 657–658.

Timothy P. Jurka, Loren Collingwood, Amber E. Boydstun, Emiliano Grossman, and Wouter van Atteveld. 2013. RTextTools: A supervised learning package for text classification. *The R J*. 5, 1, 6–12.

T. Kalamboukis and S. Nikolaidis. 1995. Suffix stripping with modern Greek. *Program: Electron. Libr. Inf. Syst.* 29, 3, 313–321.

T. Kalamboukis and S. Nikolaidis. 1999. An evaluation of stemming algorithms with modern greek. In *Proceedings of the 7th Hellenic Conference on Informatics*. 61–70.

Rohit Kansal, Vishal Goyal, and Gurpreet S. Lehal. 2012. Rule based urdu stemmer. In *Proceedings of the 24th International Conference on Computational Linguistics.* 267–276.

Nikitas N. Karanikolas. 2009. Bootstrapping the Albanian information retrieval. In *Proceedings of the 2009 4th Balkan Conference in Informatics*. 231–235.

Zied Kchaou and Slim Kanoun. 2008. Arabic stemming with two dictionaries. In *IEEE International Conference on Innovations in Information Technology*. 688–691.

Samarth Keshava and Emily Pitler. 2006. A simpler, intuitive approach to morpheme induction. In *Proceedings of 2nd Pascal Challenges Workshop*. 31–35.

Shereen Khoja and Roger Garside. 1999. Stemming arabic text. Computing Department, Lancaster University.

Shereen Khoja. 2001. APT: Arabic part-of-speech tagger. In *Proceedings of the Student Workshop at NAACL*. 20–25.

Richard Khoury and Francesca Sapsford. 2015. Latin word stemming using Wiktionary. *Digital Scholarship in the Humanities*.

Jon M. Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *J. ACM* 46, 5, 604–632.

Oskar Kohonen, Sami Virpioja, and Krista Lagus. 2010. Semi-supervised extensions to morfessor baseline. In *Proceedings of the Morpho Challenge 2010 Workshop*. 30–34.

Oskar Kohonen, Sami Virpioja, and Mikaela Klami. 2008. Allomorfessor: Towards unsupervised morpheme analysis. In *Workshop of the Cross-Language Evaluation Forum for European Languages*. Springer, 975–982.

Michal Konkol and Miloslav Konopík. 2014. Named entity recognition for highly inflectional languages: Effects of various lemmatization and stemming approaches. In *Text, Speech and Dialogue*. 267–274.

Wessel Kraaij and Renee Pohlman. 1994. Porter's stemming algorithm for dutch. In *L. G. M. Noordman and W. A. M. de Vroomen, editors, Informatiewetenschaap 1994: Wetenschappelijke Bijdragen aan de deede STINFO Conferentie.* 167–180.

Wessel Kraaij and Renee Pohlman. 1996. Viewing stemming as recall enhancement. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 40–48.

Robert Krovetz. 1993. Viewing morphology as an inference process. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 191–202.

Anand Kumar, Rajendran S, and K. P. Soman. 2014. AMRITA@ FIRE-2014: Morpheme extraction for Tamil using machine learning. In *Working Notes in Forum for Information Retrieval Evaluation (FIRE 2014)*.

Dinesh Kumar and Prince Rana. 2010. Design and development of a stemmer for Punjabi. *International J. Comput. Appl.* 11, 12, 18–23.

Mikko Kurimo, Mathias Creutz, Matti Varjokallio, Ebru Arisoy, and Murat Saraçlar. 2006. Unsupervised segmentation of words into morphemes–challenge 2005: An introduction and evaluation report. In *Proceedings of the PASCAL Challenge Workshop on Unsupervised Segmentation of Words into Morphemes*. 1–11.

Mikko Kurimo, Mathias Creutz, and Matti Varjokallio. 2008. Morpho challenge evaluation using a linguistic gold standard. In *Advances in Multilingual and MultiModal Information Retrieval, 8th Workshop of the Cross-Language Evaluation Forum for European Languages*. Springer-Verlag, Berlin, Heidelberg, 864–872.

Mikko Kurimo, Ville Turunen, and Matti Varjokallio. 2009a. Overview of morpho challenge 2008. In *Evaluating systems for Multilingual and MultiModal Information Access, 9th Workshop of the Cross-Language Evaluation Forum for European Languages*. Springer-Verlag, Berlin, 951–966.

Mikko Kurimo, Sami Virpioja, Ville T. Turunen, Graeme W. Blackwood, and William Byrne. 2009b. Overview and results of morpho challenge 2009. In *Workshop of the Cross-Language Evaluation Forum for European Languages*. Springer, 578–597.

Mikko Kurimo, Sami Virpioja, and Ville T. Turunen. 2010. Overview and results of morpho challenge 2010. In *Proceedings of the Morpho Challenge 2010 Workshop.* 7–24

Leah Larkey, Lisa Ballesteros, and Margaret E. Connell. 2007. Light stemming for Arabic information retrieval. In *Arabic Computational Morphology*. 221–243.

Leah Larkey, Lisa Ballesteros, and Margaret E. Connell. 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)*. 275–282.

Leah Larkey, Margaret E. Connell, and Nasreen Abdulijaleel. 2003. Hindi CLIR in thirty days. *ACM Trans. Asian Lang. Inf. Process*. 2, 2, 130–142.

Davor Lauc, Tomislava Lauc, Damir Boras, and Strahil Ristov. 1998. Developing text retrieval system using robust morphological parsing. In *20th International Conference on Information Technology Interfaces, ITI'98*.

Nikola Ljubešić, Damir Boras, and Ozren Kubelka. 2007. Retrieving Information in Croatian: Building a simple and efficient rule-based stemmer. *Digital Information and Heritage / Seljan, Sanja*, 313–320.

Annie Louis and Ani Nenkova. 2009. Automatically evaluating content selection in summarization without human models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 306–314.

Julie B. Lovins. 1968. Development of a stemming algorithm. *Mech. Transl. Comput. Linguist.* 11, 1, 22–31.

Sivaneasharajah Lushanthan, Ruvan Weerasingha, and D. Hearth. 2014. Morphological analyzer and generator for tamil language. In *Proceedings of the International Conference on Advances in ICT for Emerging Regions (ICTer).* 190–196.

Md. Redowan Mahmud, Mahbuba Afrin, Md. Razzaque, Ellis Miller, and Joel Iwashige. 2014. A rule based bengali stemmer. In *International Conference on Advances in Computing, Communication and Informatics*. 2750–2756.

Prasenjit Majumder, Mandar Mitra, and Dipasree Pal. 2008. Bulgarian, Hungarian and Czech stemming using YASS. In *Advances in Multilingual and Multimodal Information Retrieval*. 49–56.

Prasenjit Majumder, Mandar Mitra, Swapan K. Parui, Gobinda Kole, Pabitra Mitra, and Kalyankumar Datta. 2007. YASS: Yet another suffix stripper. *ACM Trans. Inf. Syst.* 25, 4, 18.

Mitchell P. Marcus, Beatrice Santoriniy, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn treebank. *Comput. Linguist.* 19, 2, 313–330.

James Mayfield and Paul McNamee. 2003. Single n-gram stemming. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development of Information Retrieval*. 415–416.

Paul McNamee and James Mayfield. 2004. Character n-gram tokenization for European language text retrieval. *Inform. Retriev.* 7, 1–2, 73–97.

Massimo Melucci and Nicola Orio. 2003. A novel method for stemmer generation based on hidden Markov models. In *Proceedings of the 12th International Conference on Information and Knowledge Management (CIKM'03).* 131–138.

Upendra Mishra and Chandra Prakash. 2012. MAULIK: An effective stemmer for Hindi language. *Int. J. Comput. Sci. Eng.* 4, 5, 711–717.

Christian Monson, Jaime Carbonell, Alon Lavie, and Lori Levin. 2007. ParaMor: Finding paradigms across morphology. In *Advances in Multilingual and Multimodal Information Retrieval*. Springer, 900–907.

Christof Monz and Maarten Rijke. 2002. Shallow morphological analysis in monolingual information retrieval for Dutch, German, and Italian. In *Proceedings of the Workshop of the Cross-Language Evaluation Forum for European Languages*. Darmstadt, Germany, 262–277.

Christof Monz. 2003. *From Document Retrieval to Question Answering*, Institute for Logic, Language and Computation.

Cristian Moral, Angélica de Antonio, Ricardo Imbert, and Jaime Ramírez. 2014. A survey of stemming algorithms in information retrieval. *Inf. Res.* 19, 1, 1–14.

Preslav Nakov. 2003. BulStem: Design and evaluation of inflectional stemmer for Bulgarian. In *Proceedings of Workshop on Balkan Language Resources and Tools*.

Kamal Nigam, John Lafferty, and Andrew McCallum. 1999. Using maximum entropy for text classification. In *IJCAI-99 Workshop on Machine Learning for Information Filtering*. 61–67.

Georgios Ntais. 2006. *Development of a Stemmer for the Greek Language*. Doctoral dissertation, Department of Computer and Systems Sciences, Stockholm University.

Douglas Oard, Gina-anne Levow, and Clara Cabezas. 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*. Berlin, U.K: Springer-Verlag, 176–187.

Viviane Orengo and Christian Huyck. 2001. A stemming algorithm for the Portuguese language. In *Proceedings of 8th International Symposium on String Processing and Information Retrieval*. 186–193.

Iadh Ounis, Gianni Amati, Vassilis Plachouras, Ben He, Craig Macdonald, and Christina Lioma 2006. Terrier: A high performance and scalable information retrieval platform. In *Proceedings of ACM SIGIR'06 Workshop on Open Source Information Retrieval (OSIR'06)*. 18–25.

Chris D. Paice. 1990. Another stemmer. *ACM SIGIR Forum*. 24, 3, 56–61.

Chris D. Paice. 1994. An evaluation method for stemming algorithms. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 42–50.

Jiaul H. Paik, Mandar Mitra, Swapan K. Parui, and Kalervo Jarvelin. 2011a. GRAS: An effective and efficient stemming algorithm for information retrieval. *ACM Trans. Inf. Syst.* 29, 4, 19.

Jiaul H. Paik and Swapan K. Parui. 2011. A Fast corpus-based stemmer. *ACM Trans. Asian Lang. Inf. Process.* 10, 2, 8.

Jiaul H. Paik, Dipasree Pal, and Swapan K. Parui. 2011b. 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, NY, 863–872.

Jiaul H. Paik, Swapan K. Parui, Dipasree Pal, and Stephen E. Robertson. 2013. Effective and robust query-based stemming. *ACM Trans. Inf. Syst.* 31, 4, 18.

Pratikkumar Patel, Kashyap Popat, and Pushpak Bhattacharyya. 2010. Hybrid stemmer for Gujarati. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING)*. 51.

Fuchun Peng, Nawaaz Ahmed, Xin Li, and Yumao Lu. 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, New York, NY, 639–646.

Mirko Popovic and Peter Willet. 1992. The effectiveness of stemming for natural-language access to slovene textual data. *J. Am. Soc. Inf. Sci.* 43, 5, 384–390.

Martin F. Porter. 1980. An algorithm for suffix stripping. *Program: Electron. Libr. Inf. Syst.* 14, 3, 130–137.

Martin F. Porter. 2001. *Snowball: A language for Stemming Algorithms*.

Vivek Ramachandran and Illango Krishnamurthi. 2012. An iterative stemmer for Tamil language. In *Proceedings of the 4th Asian Conference on Intelligent Information and Database Systems, ACIIDS 2012*. 197–205.

Ananthakrishnan Ramanathan and Durgesh Rao. 2003. A lightweight stemmer for Hindi. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics*. 43–48.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 133–142.

Ellen Riloff. 1995. Little words can make a big difference for text classification. In *Proceedings of 18th ACM SIGIR Conference on Research and Development in Information Retrieval*. Seattle, WA, 130–136.

Magnus Rosell. 2003. Improving clustering of Swedish newspaper articles using stemming and compound splitting. In *Proceedings of the Nordic Conference on Computational Linguistics, Reykjavik, Iceland 2003*. 1–7.

J. Sadiku. 2011. *A Novel Stemming Algorithm for Albanian Text in a Data Mining Approach for Document Classification*. Master Thesis. Tirana: University of New York Tirana.

Navanath Saharia, Utpal Sharma, and Jugal Kalita. 2012. Analysis and evaluation of stemming algorithms: A case study with Assamese. In *Proceedings of the International Conference on Advances in Computing, Communications and Informatics*. 842–846.

Shahin Salavati, Sheykh Esmaili, and Fardin Akhlaghian. 2013. Stemming for Kurdish information retrieval. In *Asia Information Retrieval Symposium*. Springer, 272–283.

Geoffrey Sampson. 2005. *The 'Language Instinct' Debate: Revised Edition*. A&C Black.

Rashmi Sankepally, Harsha Kokel, Harsha Agarwal, and Prasenjit Majumder. 2013. Morpheme extraction task at FIRE 2012–2013. In *Post-Proceedings of the 4th and 5th Workshops of the Forum for Information Retrieval Evaluation*. 5.

Jacques Savoy and Pierre-Yves Berger. 2006. Monolingual, bilingual, and GIRT information retrieval at CLEF-2005. In *Proceedings of the Workshop of the Cross-Language Evaluation Forum for European Languages (CLEF'05)*. 131–140.

Jacques Savoy. 1999. A stemming procedure and stopword list for general French corpora. *J. Am. Soc. Inf. Sci.* 50, 10, 944–952.

Jacques Savoy. 2006. Light stemming approaches for the French, Portuguese, German and Hungarian languages. In *Proceedings of the 2006 ACM Symposium on Applied Computing*. 1031–1035.

Robyn Schinke, Mark Greengrass, Alexander M. Robertson, and Peter Willett. 1996. A stemming algorithm for Latin text databases. *J. Doc.* 52, 2, 172–187.

Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *ACM Comput. Surv.* 34, 1, 1–47.

Amir A. Sharifloo and Mehrnoush Shamsfard. 2008. A bottom up approach to Persian stemming. In *Proceedings of the 3rd International Joint Conference on Natural Language Processing*. 583–588.

Manish Shrivastava and Pushpak Bhattacharyya. 2008. Hindi POS tagger using naive stemming: harnessing morphological information without extensive linguistic knowledge. In *Proceedings of International Conference on NLP (ICON'08)*.

Manish Shrivastava, Bibhuti Mohapatra, Pushpak Bhattacharyya, Nitin Agarwal, and Smriti Singh. 2005. Morphology based natural language processing tools for indian languages. In *Proceedings of the 4th Annual Inter Research Student Seminar in Computer Science*.

Jan Šnajder, Dalbelo B Bašić, and Marko Tadić. 2008. Automatic acquisition of inflectional lexica for morphological normalisation. *Inf. Process. Manag.* 44, 5, 1720–1731.

Matheus V. Soares, Ronaldo C. Prati, and Maria-Carolina Monard. 2009. Improvement on the porter's stemming algorithm for portuguese. *IEEE Latin Am. Trans.* 7, 4, 472–477.

Sebastian Spiegler, Bruno Golenia, and Peter A. Flach. 2009. Unsupervised word decomposition with the promodes algorithm. In *Proceedings of the Workshop of the Cross-Language Evaluation Forum for European Languages*. Springer, 625–632.

Sebastian Spiegler, Bruno Golénia, and Peter A. Flach. 2010. DEAP: Deductive-abductive parsing for morphological analysis. In *Proceedings of the Morpho Challenge 2010 Workshop.* 44–48.

Benno Stein and Martin Potthast. 2007. Putting successor variety stemming to work. In *Advances in Data Analysis*. Springer, 367–374.

Krzasztof Szafran. 1996. *Analizator Morfologiczny SAM-95*, Technical Report TR 96 226, Faculty of Mathematics, Informatics and Mechanics. Warsaw University.

Kazem Taghva, Rania Elkhoury, and Jeffrey Coombs. 2005. Arabic Stemming without a root dictionary. In *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05)*. 152–157.

Kristina Toutanova, Hisami Suzuki, and Achim Ruopp. 2008. Applying morphology generation models to machine translation. In *Association for Computational Linguistics*. 514–522.

Sami Virpioja, Oskar Kohonen, and Krista Lagus. 2009. Unsupervised morpheme discovery with allomorfessor. In *Working Notes for the CLEF 2009 Workshop*.

Sami Virpioja, Ville T. Turunen, Sebastian Spiegler, Oskar Kohonen, and Mikko Kurimo. 2011. Empirical comparison of evaluation methods for unsupervised learning of morphology. *Traitement Automatique des Langues (TAL)*. 52, 2, 45–90.

Justyna Wachnicka. 2004. *Odkrywanie Reguł Lematyzacji Dla j¸Ezyka Polskiego w Oparciu o Słownik Ispell-pl.*, Master's Thesis, Poznan University of Technology.

Shinji Watanabe, Tomoharu Iwata, Takaaki Hori, Atsushi Sako, and Yasuo Ariki. 2011. Topic tracking language model for speech recognition. *Comput. Speech Lang.* 25, 2, 440–461.

David Weiss. 2005. Stempelator: A hybrid stemmer for the Polish language. *Institute of Computing Science: Poznan University of Technology Research Report*.

David Weiss. 2003. Lametyzator. Retrieved from http://www.cs.put.poznan.pl/dweiss/xml/projects/lametyzator/index.xml.

Jinxi Xu and W. Bruce Croft. 1998. Corpus-based stemming using cooccurrence of word variants. *ACM Trans. Inf. Syst*. 16, 1, 61–81.

Avinash Yadav, Robins Yadav, and Sukomal Pal. 2012. ISM@FIRE-2012 adhoc retrieval and morpheme extraction task. In *Post Proceedings of FIRE-2012*.

Daniel Zeman. 2007. Unsupervised acquiring of morphological paradigms from tokenized text. In *Workshop of the Cross-Language Evaluation Forum for European Languages*. Springer, 892–899.

Daniel Zeman. 2008. Using unsupervised paradigm acquisition for prefixes. In *Proceedings of the Workshop of the Cross-Language Evaluation Forum for European Languages*. Springer, 983–990.

Imed Zitouni, Jeff Sorensen, Xiaoqiang Luo, and Radu Florian. 2005. The impact of morphological stemming on arabic mention detection and coreference resolution. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*. 63–70.

Received August 2015; revised July 2016; accepted July 2016