Cogn Comput (2017) 9:671–688
DOI 10.1007/s12559-017-9479-z

An Efficient Corpus-Based Stemmer

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

Received: 27 June 2016 / Accepted: 11 May 2017 / Published online: 7 June 2017

Abstract Word stemming is a linguistic process in which the various inflected word forms are matched to their base form. It is among the basic text pre-processing approaches used in Natural Language Processing and Information Retrieval. Stemming is employed at the text pre-processing stage to solve the issue of vocabulary mismatch or to reduce the size of the word vocabulary, and consequently also the dimensionality of training data for statistical models. In this article, we present a fully unsupervised corpus-based text stemming method which clusters morphologically related words based on lexical knowledge. The proposed method performs cognitive-inspired computing to discover morphologically related words from the corpus without any human intervention or language-specific knowledge. The performance of the proposed method is evaluated in inflection removal (approximating lemmas) and Information Retrieval tasks. The retrieval experiments in four different languages using standard Text Retrieval Conference, Cross-Language Evaluation Forum, and Forum for Information Retrieval Evaluation collections show that the proposed stemming method performs significantly better than no stemming. In the case of highly inflectional languages, Marathi and Hungarian, the improvement in Mean Average Precision is nearly 50% as compared to unstemmed words. Moreover, the proposed unsupervised stemming method outperforms state-of-the-art strong language-independent and rule-based stemming methods in all the languages. Besides Information Retrieval, the proposed stemming method also performs significantly better in inflection removal experiments. The proposed unsupervised language-independent stemming method can be used as a multipurpose tool for various tasks such as the approximation of lemmas, improving retrieval performance or other Natural Language Processing applications.

Keywords Stemming · Information retrieval · Natural language processing

Introduction

Text stemming is a basic pre-processing tool used in the fields of Natural Language Processing (NLP), Information Retrieval (IR), and Language Modeling (LM). It is a linguistic process in which the different morphological variants of the word are mapped to a common word form, also known as infinitive form. For instance, the variants, *maintained*, *maintaining*, and *maintenance*, are mapped to the word *maintain* through a stemming procedure.

Stemming has been perceived in different perspectives, and researchers in Information Retrieval and linguistic processing consider it a desirable and important step for different reasons. In Information Retrieval systems, stemming is viewed as a tool to improve retrieval accuracy by addressing the problem of vocabulary mismatch between the documents and the queries at the time of indexing and searching. So, stemming in Information Retrieval (IR) not only enhances recall but also improves precision by promoting relevant documents in superior ranks [1]. Moreover, stemming also results in a reduction of the inverted index size. Sometimes, stemming is found to be so useful that the size of the index is half of the original.

In Natural Language Processing applications, stemming is employed at the text pre-processing stage to reduce the size of the word vocabulary, and consequently also the

✉ Vishal Gupta
vishal_gupta100@yahoo.co.in

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

dimensionality of training data for statistical models. It helps in improving the performance of a number of applications such as Text Classification [2], Text Summarization [3], Machine Translation [4], POS Tagging [5], and Word Sense Disambiguation [6]. Besides these traditional applications, text stemming has also been found useful in areas like sentiment analysis and opinion mining [7, 8], e-mails, and SMS spam filtering [9]. Hence, stemming is a mechanism which is widely accepted in terms of consistency and acceptance by users.

Natural Language Processing is the core ability of cognitive computing systems. The ability to process human language naturally has led to the development of unsupervised or semi-supervised ways of stemming. In this article, we propose a cognitive computing-based stemming approach that involves self-learning of clusters of morphologically related words from the ambient corpus without any human intervention or language-specific knowledge. Our proposed method is fully unsupervised and language-independent which improves retrieval accuracy at low computation cost. The method takes a set of unique words as input and returns a set of groups of morphologically related words based on lexical knowledge between the words.

The proposed stemming algorithm works in two phases. During the first phase, we collect unique words from the document collection and group them into different classes according to common prefixes of the same length. We then proceed by calculating lexical similarity among all the word pairs in each class. In the second phase, a hierarchal clustering algorithm is used to cluster similar morphological variants based on the string distances calculated in the first phase. In order to prove the multilingual property of our algorithm, we tested the retrieval performance on four different languages, namely, English, Bengali, Marathi, and Hungarian. The languages which have been studied are of varying origin and degree of morphological complexity. Among the languages, English is least inflectional whereas Hungarian and Marathi are highly agglutinative and inflectional in nature. Bengali has moderate morphological complexity. The experimental evaluations show that the proposed algorithm provides a significant improvement in all the languages.

The rest of the article is organized as follows. The related work is presented in the "Related Work" section. In the "Proposed Unsupervised Stemming Approach" section, the proposed unsupervised corpus-based stemming approach is described. The performance evaluation of the proposed method in Information Retrieval experiments is presented in the "Information Retrieval Experiments" section. In the "Inflection Removal Experiments" section, we present the results of performance evaluation of our proposed method in inflection removal by comparing the results with manually annotated data. The "Conclusion" section concludes this article.

Related Work

Stemming is a popular technique, and a large number of stemmers of different flavors and nature are described in the literature. Currently, stemming methods can be broadly classified into two categories: *Rule-based* and *Statistical*. *Rule-based* (language-specific) stemmers make use of certain pre-defined language-related rules to map the morphological variants of the word to its base form. These language-related rules are created manually by the language experts or linguists. A number of language-specific stemmers are available for English and European languages but for many other languages developing linguistic rule-based stemmers require some additional resources which are either incomplete or unavailable. *Statistical* stemmers, on the other hand, preclude the need for any additional resources or language expertise. These techniques incorporate new language into the system with very little effort, and this is useful especially for applications related to Information Retrieval. Moreover, these techniques can deal with the languages that have complex morphology and sparse data. In the following subsections, we review some rule-based and statistical methods.

Rule-Based Methods

For stemming in English, Lovins [10] was the first stemming algorithm published in the literature. It makes use of a lookup table to remove endings on the basis of the longest match principle. Dawson [11] proposed an extended and improved version of the Lovins stemmer by providing a more exhaustive list of nearly 1200 endings. Although the Lovins stemmer was the first to be published, Porter's stemmer [12] is the most popular, likely being widely used in IR tasks due to the balance between efficiency and simplicity. The Porter stemmer is a non-recursive rule-based stemmer which makes use of nearly 60 rules that are applied successively in five steps. A number of other rule-based stemmers exist for English such as the Paice/Husk stemmer [13] or Krovetz [6].

Although stemming in English has showed mixed results, it has been found to be quite useful in languages with complex morphology where a single stem can have a large number of variants, such as Roman [6, 14], Dutch [15], Finnish, Hungarian [16, 17], Greek [18], or Czech [19, 20] languages.

Statistical Methods

Statistical stemmers are based on unsupervised or semi-supervised learning of morphology of the language from the 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. Depending upon the nature of the algorithm, statistical techniques can be further classified into the following two categories:

Lexicon Analysis-Based Approaches These approaches simply analyze the words in the corpus to group morphologically related words. These approaches look at each word individually and conflate variant word forms using various methods such as computation of frequency of substrings and string distances.

Co-occurrence-Based Approaches These methods attempt to analyze words on the basis of their context or co-occurrences with the other words in the corpus. These methods are based on the assumption that the words which co-occur are much better members to be merged in a class than the words which do not co-occur [21]. Co-occurrence-based approaches require relatively larger corpora as compared to lexicon analysis-based methods so as to increase the reliability of the co-occurrence data [22].

In the subsequent subsections, various lexicon analysis- and co-occurrence-based stemmers, proposed in the literature, are discussed.

Lexicon Analysis-Based Approaches

Orad et al. [23] developed a stemming technique which discovers suffixes from the corpus statistically and removes them from the word endings. The frequency of every ending of lengths 1, 2, 3, and 4 that produces a root of length at least three characters is computed from the first 500,000 words in the corpus. The frequencies are then adjusted by subtracting the frequency of subsuming ending from the next longer-length suffix. The adjusted frequencies are then arranged in decreasing order for each length and are used to determine the most likely valid suffixes.

An unsupervised stemming technique using a minimum description length (MDL) approach is described in Goldsmith [24]. The technique proposed in the paper calculates the frequency of the root and suffix at every possible breakpoint for each word in the collection. The breakpoint is optimal if for every instance of the word the breakpoint is the same and minimizes the necessary bits required to encode the text collection (which is analogous to reducing the entropy of the collection). The software framework for this approach is named as Linguistica [25].

Melucci and Orio [26] applied the concept of the hidden Markov model (HMM) to stemming. The sequence of letters forming a word in a language is represented by the sequence of symbols emitted by an HMM at each transition. The transition and emission probabilities are computed through training on a set of words, based on the Baum-Welch expectation-maximization (EM) method. There can be more than one path or state sequence for a single word. For each input word, the most probable path is computed using the Viterbi algorithm and the transition from a state belonging to the prefix set to a state belonging to the suffix set in the most probable path is the *split-point*. The sequence of letters before the *split-point* is considered as the stem of the word. In case there is no *split-point*, then the whole word is considered as the stem with no suffix. The HMM-based stemming method performed equivalently to the rule-based stemmers in a number of European languages.

Bacchin et al. [27] suggested a probabilistic model for unsupervised stemmer generation. The proposed technique is an extension of the author's previous work on graph-based stemming described in Bacchin et al. [28], as it models the mutual reinforcement between the stem and the derivation using the probabilistic framework. The method first discovers a set of substrings by splitting each word in the corpus at every possible location. The set of the substrings is then converted into a directed graph whose nodes are the substrings, and there is an edge between nodes x and y if they induce a word w = xy. The prefix and suffix scores are then computed from the graph using the Hyperlink-Induced Topic Search (HITS) link analysis algorithm based on the presumption that good prefixes point to good suffixes, and good suffixes are pointed to by good prefixes. The prefix and suffix scores are used to find the best split of the word by maximizing the probability of the prefix and suffix pair. The authors claimed that the performance of the stemmer based on the probabilistic model is as effective as stemmers based on linguistic knowledge in Information Retrieval tasks.

Creutz and Lagus [29] developed a probabilistic model family, named *Morfessor*, for unsupervised segmentation of words and learning of morphology of the language from the corpus of unannotated text. The Morfessor model was developed in a probabilistic maximum a posteriori framework and is suitable for highly inflectional languages where words can be formed by a large sequence of morphemes. The model aims to discover a representation of the data that is as compact as possible by finding an optimal balance between compactness of the corpus representation and the compactness of the morph lexicon. The first version of Morfessor model called *Morfessor Baseline* has been introduced in Creutz and Lagus [30] whose formulation is followed from the "recursive minimum description length (MDL)" principle. The baseline model is a context-independent model that discovers optimal segmentation of the source words by minimizing an MDL model cost function. The total model cost considers the cost of representation of both the source text and the morph vocabulary. The model produces flat morphs

consisting of a string of letters and never has a hierarchal structure. A slight modification of the Morfessor baseline model called *Morfessor Baseline-Length* has been described in Creutz [31]. It is similar to the Baseline model except that it makes use of gamma distribution for modeling the length of morphemes. Both Baseline and Baseline-Length models perform equally well in a morpheme segmentation task for large-sized corpora [29].

Two extended versions of Morfessor, namely, *Morfessor Categories-Maximum Likelihood* (ML) [32] and *Morfessor Categories-Maximum* a Posteriori (MAP) [33], introduce morph categories. The morphs are categorized as *prefix*, *stem*, *suffix*, or *noise* (non-morpheme). Both these category-based models refine the initial segments produced by the Baseline model to alleviate common errors such as over-segmentation of rarely occurring strings and under-segmentation of commonly occurring strings. In the Morfessor Category-ML model, the optimal segmentations are modeled using a hidden Markov model with emission likelihoods of morphs from categories and transition likelihoods between categories. Morfessor Category-MAP is the most extensive model and reformulates the Category-ML model in a more elegant way. The main distinguishing feature of the Category-MAP model is that morphs can have hierarchal structures, i.e., a morph either consists of a string of letters or two or more submorphs which help in reducing under-segmentation and over-segmentation errors as the nodes in the tree are not expanded if the next level has non-morphemes.

Kohonen et al. [34] also proposed an extension to Morfessor called *Allomorfessor* which employs MAP estimations for optimal segmentation of the source text. It discovers a common base form for allomorphs (morphemes which have several base forms) in the corpus. It is based on the principle that the base forms of various morphemes are similar, and the changes occur near the boundary. So, the various allomorphs are modeled by mutations. However, the method did not perform well in Information Retrieval and Machine Translation tasks.

Conflation is viewed as a word clustering problem in Majumder et al. [35] to develop a language-independent stemmer named Yet Another Suffix Stripper (YASS). The method makes use of certain string distances that do not have any knowledge of morphological variants of the language. After calculating the string distance parameters, the clusters are obtained using a graph-based complete linkage technique. The string distance used in this technique is inappropriate for long-length suffixes. YASS is therefore found to be less effective on agglutinative languages like Marathi and Hungarian where words with long-length suffixes are quite common. Our method uses the well-known Jaro-Winkler distance metric [36, 37] to group morphologically related words, which is found to be suitable for all kinds of suffixing languages. It awards longest common prefixes between two-word pairs and considers insertions, deletions, and substitutions of characters while comparing the words. The Jaro-Winkler metric has been found to be quite useful and is extensively used for approximate string matching in a number of applications such as cognate identification [38], name matching, and record linkage [39–41].

Paik et al. [42] used weighted graphs to build a stemmer named GRAph-based Stemmer (GRAS). The stemmer discovers frequently occurring suffix pairs, rather than single suffix from the words in the corpus. A weighted undirected graph is then constructed using the knowledge of the suffix pairs whose nodes are the words of the corpus, and there is an edge from node x to y if both words induce a suffix pair whose frequency is above some pre-decided threshold. The graph is then decomposed by computing cohesion between the nodes.

Paik and Parui [43] proposed a stemmer that performs stemming on the basis of potential suffixes discovered from the corpus. In this technique, the potential suffixes are first identified according to their frequency in the corpus. The words are then clustered into classes according to common prefix, and the strength of each class is measured using potential suffix knowledge. The strength determines whether the common prefix string of the class is the stem or not. If not, then another root in the class is determined iteratively.

Co-occurrence-Based Approaches

Xu and Croft [1] first used a co-occurrence-based approach to refine or to develop a stemmer. The authors used a variation of the Expected Mutual Information Measure (EMIM) to calculate the association between the word pairs. After calculating the EMIM scores, the equivalence classes already created by some aggressive stemmer like Porter are upgraded using connected component and/or optimal partitioning graph clustering algorithms. A similar kind of approach is proposed by Paik et al. [21] which makes use of a simpler co-occurrence measure and a novel nearest neighbor-based approach.

Bhamidipati and Pal [2] proposed a technique for stemmer refinement that considers classification knowledge of the corpus. The words are stemmed on the basis of their distribution similarity over various document categories where the distribution of each word is calculated from the frequency of occurrence in all the categories.

Peng et al. [44] pointed out that blind stemming during query expansion does not consider the context in which the term is used and performs blind comparisons of all the occurrences in the documents. To overcome these

problems, a context-sensitive stemming approach is proposed that performs context-sensitive stemming both at the query and at the document side.

Paik et al. [22] proposed a query-based stemming technique that provides those variants which are thematically coherent with the words of the query. Brychcín and Konopík [45] proposed an unsupervised stemming algorithm which works in two phases. In the first phase, the words are grouped on the basis of lexical and semantic knowledge using a modified minimal mutual information (MMI) clustering algorithm. In the second phase, the clusters generated in the first phase are used by the maximum entropy classifier to decide when and how to stem the input word. The authors experimentally verified that the stemmer increases precision by providing accurate stems at the expense of a very little decrease in recall.

Character N-Gram-Based Indexing

Character n-gram indexing is an alternate approach for building retrieval systems that avoid the requirement of having proper tokenization and stemming procedures. In the character n-gram indexing scheme, n-grams (sequence of n adjacent characters) instead of words are used as indexing units for Information Retrieval systems. McNamee et al. [46] tested character n-gram indexing in a number of languages and reported that 4 or 5-gram is effective in Information Retrieval systems. Pirkola et al. [47] and Järvelin [48] used s-grams for Information Retrieval applications. In the s-gram technique, a varying number of characters may or may not be skipped while forming n-length substrings. n-grams are a special case of s-grams when no characters are skipped while forming substrings with n characters. The major limitation of n-gram or s-gram indexing is the increase in the inverted index size. A sequence of m characters has m − n + 1 n-grams of length n but only (m + 1) / (k + 1) words, where k is the average length of the word for the language. Due to the increase in the size of the index, query execution and retrieval time also increase substantially. Character 4-gram-based retrieval is approximately ten times slower as compared to word retrieval [43]. Various studies [19, 20, 49] suggested that stemming approaches are more effective as compared to character n-gram indexing for handling the morphological variations in Information Retrieval systems.

Proposed Unsupervised Stemming Approach

The major aim of our work is to propose a fully unsupervised language-independent stemming algorithm that can serve as a multipurpose tool in a number of applications. The proposed algorithm considers unique words collected from the corpus as input and returns a set of equivalence classes, where each class contains morphologically related words. Our algorithm works in two phases as follows:

1. Determine string distance between word pairs.
2. Group the words based on their distance information.

Measuring String Distances

In this phase, string distances between the word pairs are computed. The string distance function is used to map two-word pairs to a number, where a high value of this number indicates less similarity between the word pairs. We start with a lexicon containing unique words collected from a corpus. The lexicon is divided into a number of classes such that each class contains words which share a common prefix of the same length. We chose common prefix length as 3 so that variants of words with small length like *eat* and *eats* are not ignored. The partitioning of the lexicon into equivalence classes of common prefix will help in reducing the computational load of the algorithm as rather than computing distance between all the word pairs of the lexicon, distance between word pairs belonging to the same class is computed. In our approach, we used the well-known Jaro-Winkler distance to find similarity between the morphologically related words. It is defined as follows:

Jaro-Winkler Distance The Jaro-Winkler distance is a variation of the Jaro string distance [36, 50]. The Jaro distance is a type of edit distance that considers insertions, transpositions, and deletions for comparing two strings. The Jaro metric calculates the lengths of the strings, the number of common characters in the strings, and number of transpositions. The Jaro similarity between "common word string" (w_1) and "test string" (w_2) of length l_1 and l_2 respectively, is given by

$$\text{Jaro-Sim}(w_1, w_2) = \Phi = \frac{1}{3}\left(\frac{c}{l_1} + \frac{c}{l_2} + \frac{c-t}{c}\right) \quad (1)$$

c = number of common characters between strings w_1 and w_2, where common characters should not be farther than $\left\lfloor \frac{\max(l_1,\ l_2)}{2} \right\rfloor - 1$.

t = number of transpositions, where transpositions are defined as the number of matching characters (but different sequence order) divided by 2.

Winkler [37] slightly modified Eq. (1) by refining the weights for poor matching word pairs and rewarding longest common prefix as

$$\text{JaroWinkler-Sim}(w_1, w_2) = \Phi_{jw} = \Phi + \text{L.p.}(1-\Phi) \quad (2)$$

where L is the length of the common prefix and p is a scaling factor taken as 0.1. The Jaro-Winkler distance between two strings is therefore computed as $1 - \Phi_{jw}$.

The length of the common prefix is restricted to four characters in the original implementation. However, we did not restrict the length of the common prefix to 4 and considered the actual length of the common prefix so as to award the longest common prefix and avoid early mismatch among the pair of words. It may result in the value of the similarity function shown in Eq. (2) to be greater than 1 for long-length strings, and then the conversion of the similarity score to a distance value would possibly return a negative value. But this is not a problem for the clustering algorithm described in the next phase. In fact, no restriction on the length of the common prefix helps in grouping the long-length variant word forms in the same equivalence class thereby making the string distance more effective on highly inflectional languages.

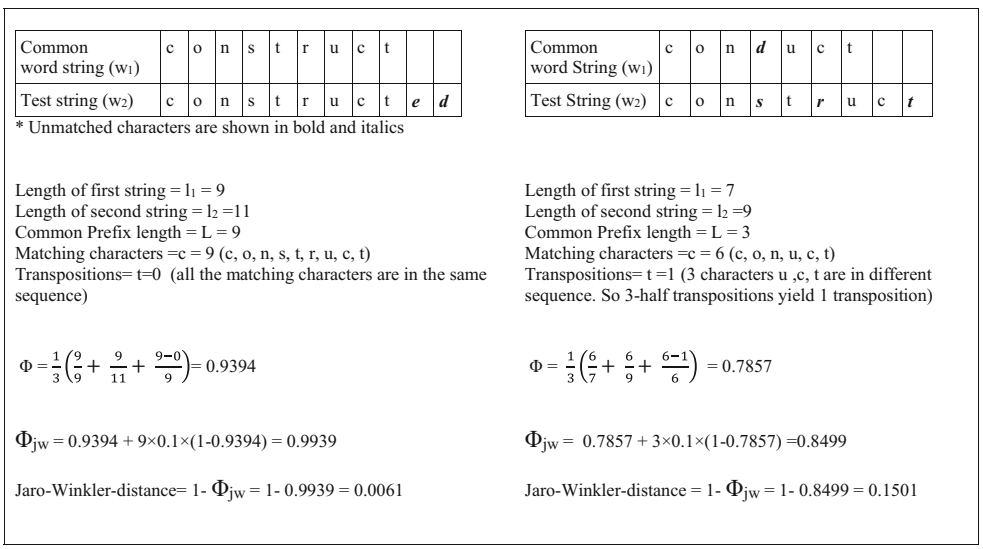
Figure 1 presents the calculation of the Jaro-Winkler distance between the word pairs (*construct*, *constructed*) and (*conduct*, *construct*). It is clear from the values that the words *conduct* and *construct* are farther apart than the words *construct* and *constructed*.

ALGORITHM 1: Measuring String Distance

1. Partition the lexicon into a number of equivalence classes C_1, C_2, C_3, ... C_n such that each class contains words which share a common prefix of length 3.
2. for each class Ci= {w_1, w_2, w_3, ..., w_n} **do**
3. for any two-word pairs (w_j, w_k) ε C_i (j<k) **do**
4. Compute the Jaro-Winkler distance (w_j, w_k).
5. **end for**
6. output triangular distance matrix D_i for each class.
7. **end for**

Grouping the Words

The distance between the word pairs defined in the previous phase is used to group morphologically similar words. All the words within the group are represented by the central word, i.e., longest common substring of that group. We employed a hierarchical agglomerative clustering (HAC) algorithm due to its ability to create natural clusters, possibility to view data at different threshold levels and no prior knowledge of number of clusters. A number



Common word string (w_1)	c	o	n	s	t	r	u	c	t		
Test string (w_2)	c	o	n	s	t	r	u	c	t	***e***	***d***

* Unmatched characters are shown in bold and italics

Length of first string = l_1 = 9
Length of second string = l_2 =11
Common Prefix length = L = 9
Matching characters =c = 9 (c, o, n, s, t, r, u, c, t)
Transpositions= t=0 (all the matching characters are in the same sequence)

$\Phi = \frac{1}{3}\left(\frac{9}{9} + \frac{9}{11} + \frac{9-0}{9}\right) = 0.9394$

$\Phi_{jw} = 0.9394 + 9\times0.1\times(1-0.9394) = 0.9939$

Jaro-Winkler-distance= 1- Φ_{jw} = 1- 0.9939 = 0.0061

Common word String (w_1)	c	o	n	***d***	u	c	t		
Test String (w_2)	c	o	n	***s***	t	***r***	u	c	***t***

Length of first string = l_1 = 7
Length of second string = l_2 =9
Common Prefix length = L = 3
Matching characters =c = 6 (c, o, n, u, c, t)
Transpositions= t =1 (3 characters u ,c, t are in different sequence. So 3-half transpositions yield 1 transposition)

$\Phi = \frac{1}{3}\left(\frac{6}{7} + \frac{6}{9} + \frac{6-1}{6}\right) = 0.7857$

$\Phi_{jw} = 0.7857 + 3\times0.1\times(1-0.7857) = 0.8499$

Jaro-Winkler-distance = 1- Φ_{jw} = 1- 0.8499 = 0.1501

Fig. 1 Calculation of string distance between word pairs

of articles have presented hierarchical clustering procedures for grouping words in a corpus, such as the well-known Brown clustering method [51] which groups words into clusters based on distributional information. The hierarchical agglomerative clustering algorithm begins by considering each word as a separate cluster, so for N words in a class, there are N clusters. At each next step, the algorithm merges the most similar clusters and updates the distance between the new cluster and each old cluster. The process continues until the distance between the clusters is less than the pre-defined threshold value. At each fusion stage of clusters, a linkage method decides how the inter-cluster distances are calculated. A number of linkage rules like *single linkage*, *complete linkage*, and *average linkage* are described in the literature [52]

The *single linkage* (also called nearest neighbor or minimum) method employs a friends of friends clustering method. In this method, the distance between two clusters is equal to the least distance between an object in one cluster and an object in another cluster. The single linkage clustering method tends to produce long and elongated chain-like clusters when the objects are quite close to each other. The clusters produced by the single linkage method are internally heterogeneous, but the objective of clustering is to produce homogeneous clusters. In the *Complete linkage* (also called farthest neighbor or maximum) method, the distance between the clusters is equal to the greatest distance between the object in one cluster and an object in another cluster. The complete linkage method tends to produce more balanced and compact clusters with equal diameters as compared to the single linkage method. The *average linkage* method is an attractive trade-off between single and complete linkage methods where the distances are averaged at every amalgamation step. In the average linkage method, the distance between two clusters is equal to the average distance between each object in one cluster to every object in the other cluster. The average linkage method employs a more central measure as compared to single and complete linkage methods (as they use single pairwise distance). So, we chose the average linkage method in our experiments.

ALGORITHM 2: Grouping the Words

1. for each distance matrix, D_i **do**
2. find the pair of objects with the least distance i.e. least dissimilar pair and merge both the objects into a single cluster.
3. update distance matrix by calculating the distance between clusters as

$$D_{\text{avg}}(C_i, C_j) = \frac{1}{|C_i| \times |C_j|} \sum_{x \in C_i, y \in C_j} d(x, y)$$

4. repeat steps 2 and 3 until all the distances in the distance matrix are less than a specified threshold θ.
5. for each cluster obtained, stem all the words in the cluster to the longest common substring of the cluster.
6. **end for**

Information Retrieval Experiments

In order to assess the effectiveness of our proposed method, we tested the retrieval performance of our approach along with other existing approaches on standard test collections. We followed the following steps for each language.

1. Find all the unique words after eliminating the numbers and the stopwords from the corpus.
2. Group words into classes and compute the distance matrix for each class according to algorithm 1.
3. Find equivalence classes of words using algorithm 2.
4. Perform retrieval experiments using the classes formed by the proposed approach.
5. Compare the performance of the proposed approach with other popular rule-based and statistical stemmers.

In the subsequent subsections, we describe the corpora, evaluation metrics, experimental system, baseline stemmers, and the experimental results.

Description of Corpora

The retrieval experiments have been conducted in four languages, namely, English, Marathi, Bengali, and Hungarian,

Table 1 Statistics of the corpora used for IR experiments

Language	Source	No. of documents	Mean words/ document	Corpus size (in MB)	No. of queries	No. of relevant documents
English	TIPSTER disks 1 and 2	173,252	337	533.3	100	8469
Marathi	FIRE 2010	99,275	273	491.3	75	1550
Hungarian	CLEF 2004–2008 Test Suite	49,530	142	105	150	3158
Bengali	FIRE 2010	123,047	362	774.1	125	2373

using standard Text Retrieval Conference (TREC), Cross-Language Evaluation Forum (CLEF), and Forum for Information Retrieval Evaluation (FIRE) collections. Among these languages, English has the simplest morphology with comparatively less number of inflections whereas Hungarian and Marathi are highly inflectional languages. Table 1 summarizes the statistics of the corpora used in the experiments. Each collection comprises queries which are structured according to TREC ad hoc track specifications. The queries consist of three sections: *title*, *description*, and *narrative*. The title section contains few words representing the user query submitted to the search engine. The description section represents a description of the requested data in one or two sentences, and the narrative section contains few sentences expressing the condition of relevance for the requested data. We conducted all the retrieval experiments using the title and description sections of the query. The details of the document collection and query sets for each language are as follows:

English English experiments have been carried out on *Wall Street Journal* document collection provided by TIPSTER disks 1 and 2. The collection consists of 173,252 documents with an average of 337 words per document. We performed retrieval experiments using the TREC 2–3 ad hoc topics containing 100 (101–200) queries.

Marathi The Marathi document collection has been taken from FIRE[1] 2010 containing *Maharashtra Times* and *Sakal* news articles over a span of 4 years from 2004 to 2007. The collection consists of 99,275 documents with an average of 273 words per document. The retrieval experiments have been conducted using 100 (26–125) FIRE ad hoc queries.

Hungarian The document collection and query sets for Hungarian have been taken from CLEF[2] ad hoc test suites. The corpus is the Magyar Hirlap 2002 collection containing 49,530 documents with an average of 142 words per document. The retrieval experiments are performed using 150 queries.

Bengali The Bengali document collection has also been taken from FIRE 2010 containing *Bengali Daily Anandabazar Patrika* news articles from 2004 to 2007. The collection contains 123,047 documents with an average of 362 words per document. The corpus is supplemented by 125 (26–150) FIRE ad hoc queries.

Experimental System and IR Model

Indexing and retrieval experiments are performed using the TERRIER[3] Information Retrieval system. TERRIER is an open-source retrieval system written in Java, developed at the School of Computing Science of the University of Glasgow. TERRIER is a comprehensive, efficient, and flexible platform for development and evaluation of large-scale retrieval applications.

TERRIER provides support for state-of-the-art retrieval approaches such as tf-idf, BM25, Divergence from Randomness (DFR), and language modeling. We used the IFB2 retrieval model [53] from the DFR paradigm in our experiments. The authors of this weighting model compared a number of retrieval models, and IFB2 is reported to be one of the best models. We also compared several retrieval models in our previous article [54] and suggested IFB2 to be one of the best-performing models. The IFB2 model is a combination of term frequency and inverse collection term frequency with two-level normalization. It uses inverse term frequency as basic randomness model with Bernoulli after-effect (ratio of two Bernoulli processes) as the first normalization and normalization2 (term frequency density is decreasing function of document length) as term frequency normalization. The first level of normalization considers the risk factor involved in considering a term as a good descriptor for the document. It is based on the assumption that the gain is directly related to the risk involved in considering the term as a good descriptor for the possibly relevant document. The second level of normalization adjusts the frequency of the term according to the document length by hypothesizing that the term frequency density function is the decreasing function of the length of the document. The documents in the collection are indexed on the basis of unigram word model assumption, and the

[1] http://www.isical.ac.in/~fire
[2] http://clef-campaign.org/
[3] http://terrier.org/

query is represented as a set of different terms. The ranking function for the IFB2 retrieval model is given by Eq. (3)

$$\text{Score}(q,d) = \sum_{t \in q} \text{QTF}(t).\text{NTF}.\frac{F+1}{\text{DF}(\text{NTF}+1)}.\log_2\frac{N+1}{F+0.5} \quad (3)$$

$$\text{NTF} = \text{term\_freq}.\log_2\left(1+\frac{\text{mean\_doc\_length}}{\text{doc\_length}}\right) \quad (4)$$

where QTF and NTF denote query term frequency and normalized term frequency, respectively; DF denotes document frequency of term t. F and N represent collection term frequency and collection size respectively.

Baseline Methods

We compared the performance of our proposed method with the strong state-of-the-art methods and also against no stemming. The baseline stemmers have been selected from rule-based, lexicon analysis-based, and co-occurrence-based categories. Specifically, we considered the following baseline methods in our experiments:

Rule-Based Stemmers In the case of English, we used well-known Lovins [10] and Porter [12] stemmers. For Marathi and Bengali, the light stemmers[4] described in Dolamic and Savoy [49] have been used. For Hungarian, a Porter style stemmer obtained through the Snowball framework has been used.

LINGUISTICA An unsupervised morphological analyzer[5] presented in Goldsmith [24] based on the minimum-description-length model has been used. This stemmer does not require any parameter tuning, but the number of tokens is set to 15,000,000 according to the available implementation of the stemmer.

MORFESSOR The recursive MDL-based unsupervised data-driven method for morpheme segmentation has been used. We used the baseline model described in Creutz and Lagus [30] which does not require any parameter tuning. The results have been produced by the code[6] made available by the authors under GNU Public License.

4-GRAM We also compared our proposed stemming method against a character n-gram indexing scheme [46] which uses n-grams instead of words as indexing terms. The value of n is taken as 4 as suggested by the authors.

Yet Another Suffix Stripper A clustering-based unsupervised method described in Majumder et al. [35] has been used. The results of the YASS have been produced using the author's code.[7] The threshold for the clustering algorithm has been set to 1.5 according to the recommendation of the authors.

Graph-Based Stemming Graph-based word clustering using suffix pair frequency described in Paik et al. [42] has been used. The threshold for frequent suffix pairs was set to 4, and the cutoff for cohesion (to determine whether two nodes are morphologically related or not) was set to 0.8 for all the languages. These parameters have been suggested by the authors.

High-Precision Stemmer An unsupervised stemmer [45] that uses lexical and semantic information of words has been used. The results are produced using the code[8] made available by the authors. The maximum suffix length was set to 3, and the minimum lexical distance between words with the same stem was set to 0.7 for English and Bengali and 0.6 for Marathi and Hungarian according to the recommendation of the authors.

Evaluation Metrics

The retrieval performance of the various stemmers has been compared using Mean Average Precision as the primary metric. In order to see the increase in retrieval precision, precision@10 (P@10) and R-Precision values have also been reported. Stemming is often regarded as a recall enhancing method, and so the number of relevant documents retrieved is also reported for all the languages. The values in the brackets in each table denote the percentage increase or decrease in the values as compared to no stemming.

The query-wise performance in terms of average precision values has also been compared in each language. The number of queries for which average precision values are more or less as compared to no stemming has been reported. The Robustness Index (RI) (given in Eq. 5) as used by Sakai et al. [55] and Paik et al. [22] has also been calculated. It measures the number of queries improved and the extent of improvement. But one limitation of this metric is that it does not consider the magnitude of differences. So, the query-wise average precision values are also statistically compared against no stemming using paired *t* test at the 95% confidence level.

$$\text{Robustness Index} = \text{RI} = \frac{N^+ - N^-}{|Q|} \quad (5)$$

where N^+ and N^- denote the number of queries with average precision values greater and lesser than no stemming, respectively.

[4] Available at http://members.unine.ch/jacques.savoy/clef/index.html
[5] Available at http://linguistica.uchicago.edu/linguistica.html
[6] Available at http://www.cis.hut.fi/projects/morpho/
[7] Available at http://www.isical.ac.in/~clia/resources.html
[8] Available at http://liks.fav.zcu.cz/HPS/

Table 2 Effect of threshold (θ) on MAP values for training queries

English		Marathi		Hungarian		Bengali	
Threshold	MAP	Threshold	MAP	Threshold	MAP	Threshold	MAP
0.08	0.3317	0.18	0.4407	0.18	0.3336	0.18	0.4680
0.09	0.3400	0.19	0.4480	0.19	0.3397	0.19	0.4719
0.1	**0.3459**	0.2	**0.4534**	0.2	0.3420	0.2	**0.4789**
0.11	0.3417	0.21	0.4505	0.21	**0.3489**	0.21	0.4695
0.12	0.3387	0.22	0.4474	0.22	0.3377	0.22	0.4656

The best MAP values for all the languages are marked in bold

Parameter Setting

Our proposed unsupervised approach depends on only one corpus-based parameter, i.e., a threshold θ above which the average linkage clustering algorithm terminates, and no further merging of clusters takes place. The value of this threshold should be carefully chosen as a low value of the threshold would result in a lenient stemmer that will be unable to group all morphological variants in the same group whereas a high value of threshold would result into an aggressive stemmer that tends to overstem the words. So, to decide an appropriate value for this threshold, we performed retrieval experiments at various threshold levels in all the four languages. In order to avoid any bias, we used different queries for learning parameter (this section) and testing ("Retrieval Results" section). For learning parameters, we used 50 queries each for English (TREC 151–200), Bengali (FIRE 76–125), Marathi (FIRE 76–125), and Hungarian (CLEF 2005). In our experiments, we found that maximum improvements in MAP values for English (least inflectional language) were achieved for the threshold between 0.08 and 0.12 while for other languages the maximum improvements in MAP values are obtained for the threshold between 0.18 and 0.22. The retrieval results for all the languages at the various threshold levels are presented in Table 2.

The bold values in Table 2 indicate that the best MAP is achieved at threshold 0.1 for English, 0.2 for Bengali and Marathi, and 0.21 for Hungarian. So, we used θ = 0.1 for English and θ = 0.2 for Marathi, Hungarian, and Bengali.

Retrieval Results

In this section, we present the retrieval results of the proposed method and baseline rule-based and language-independent stemmers. In each language, two tables are shown: the first table presents the retrieval results of stemmers in terms of MAP, P@10, R-P, and the number of relevant documents retrieved, and the second table presents the query-wise performance and *p* values of the statistical tests of various stemmers against no stemming. The values in bold denote the best performance in the specific category.

English Results

English retrieval experiments are performed on 173,252 WSJ documents using 100 topics (TREC queries 101–200). The insignificant stopwords are removed using a standard stopwords list available in the TERRIER system containing 733 words. Table 3 shows the retrieval results on English test collection. It is clear from the table that all the methods performed better than no stemming. The proposed method and the Porter stemmer performed almost equal and achieved an improvement of 10.41 and 10.30%, respectively, in MAP against unstemmed words. GRAS also performed fairly well and reported an improvement of 9.11% in MAP as compared to no stemming. Lovin, YASS, MORFESSOR, and HPS performed almost equally and increased MAP by nearly 7%. However, 4-GRAM indexing and LINGUISTICA stemming approach performed inferior as compared to other methods and managed to show an improvement of 3.38 and 4.57%, respectively, in MAP as compared to the unstemmed words.

The query-by-query analysis presented in Table 4 reveals that both the proposed method and the Porter stemmer improved the average precision of 63 queries out of 100 against no stemming. The *p* values in Table 4 indicate that all the methods, except 4-GRAM indexing and LINGUISTICA, performed statistically better than no stemming. The proposed method gave the best statistical performance as compared to no stemming.

Marathi Results

Marathi is a highly inflectional Indian language. Due to the morphological richness of the language, we decided to evaluate our approach on it. Marathi retrieval experiments are performed on 99,275 documents using 75 FIRE topics. The stopwords are removed using a standard stopwords list[9]

[9] Available at http://members.unine.ch/jacques.savoy/clef/marathiST.txt

Table 3 Retrieval results for English

Total relevant documents: 8469 number of queries: 100 (TREC 101–200)				
Method	MAP	R-P	P@10	Rel. Ret.
NO STEM	0.2931	0.3286	0.4970	5956
LOVIN	0.3136 (7.00)	0.3440 (4.69)	0.5105 (2.72)	6351 (6.63)
PORTER	0.3233 (10.30)	0.3487 (6.12)	0.5010 (0.80)	6321 (6.13)
LINGUISTICA	0.3065 (4.57)	0.3393 (3.26)	0.4980 (0.20)	6205 (4.18)
MORFESSOR	0.3131 (6.82)	0.3461 (5.32)	0.5030 (1.21)	6263 (5.15)
4-GRAM	0.3030 (3.38)	0.3307 (0.64)	0.4900 (−1.41)	6193 (3.98)
YASS	0.3134 (6.93)	0.3433 (4.47)	0.5000 (0.60)	6327 (6.23)
GRAS	0.3198 (9.11)	0.3500 (6.51)	0.5090 (2.41)	6306 (5.87)
HPS	0.3127 (6.69)	0.3398 (3.41)	0.5110 (2.82)	6335 (6.36)
PROPOSED	**0.3236 (10.41)**	**0.3503 (6.60)**	**0.5120 (3.02)**	**6356 (6.72)**

The best retrieval performance is marked in bold

containing 99 words. The retrieval results for Marathi are presented in Table 5. Unlike English, stemming is found to be more advantageous in Marathi. The proposed method showed large improvement of 47.5% in MAP as compared to the unstemmed words. GRAS and MORFESSOR also performed well and reported an improvement of nearly 42% in MAP. Unlike English, 4-GRAM indexing and LINGUISTICA performed consistently well in Marathi and achieved an improvement of 36.6 and 33.2%, respectively, in MAP as compared to no stemming. HPS and YASS also performed fairly well as compared to the unstemmed run. Interestingly, rule-based Marathi stemmer performed worse among all the stemmers under analysis and reported a marginal improvement of nearly 3% in MAP as compared to no stemming.

Table 6 shows that the proposed unsupervised approach outperforms all the methods on a substantial number of queries. The proposed method improved the average precision of 61 queries out of 75 as compared to the unstemmed run. The performance difference of various methods against no stemming (except rule-based) is found to be statistically significant. It is clear from the p values presented in Table 6.

Hungarian Results

Like Marathi, Hungarian is also highly inflectional and has a large number of compound words. The retrieval experiments of Hungarian are performed on Magyar Hirlap collection of 49,530 documents using 150 topics. The stopwords are removed using a stopwords list[10] containing 737 words. The retrieval results of Hungarian are presented in Table 7. In Hungarian, all the methods reported large improvements as compared to the no stemming baseline in terms of both mean average precision and total number of relevant documents retrieved. GRAS and the proposed method performed almost equal and showed an improvement of nearly 50% in MAP. Rule-based Snowball stemmer, YASS, MORFESSOR, and HPS also performed almost equally and reported an improvement of nearly 45% against the unstemmed run. 4-GRAM and LINGUISTICA did fairly well and showed an improvement of 26.7 and 24.2%, respectively, in MAP as compared to no stemming. Out of 3158 relevant documents, the proposed method and GRAS retrieved 2689 documents thereby showing an increase of almost 37%.

Table 8 presents query-by-query analysis and statistical significance results for the Hungarian collection. It is clear from the table that results of all the stemming methods are statistically significant as compared to the unstemmed run. The proposed unsupervised approach improved a maximum number of 107 queries out of 150 and achieved a high Robustness Index of 0.44.

Bengali Results

The retrieval experiments on Bengali are performed on FIRE 2010 collection containing 123,047 documents and 125 queries (FIRE 26–150 topics). The insignificant Bengali stopwords are removed using a standard stopwords list[11] containing 114 words. Unlike Hungarian and Marathi, stemming is found to be less beneficial for Bengali. However, the stemming results of most of the methods are found to be statistically significant as compared to the unstemmed run. Table 9 presents retrieval results of various stemmers on Bengali test collection. We can see from the table that the proposed method performed best among all the methods. The proposed method achieved the highest improvement of nearly 14% in MAP against the unstemmed word retrieval. HPS and GRAS also performed well and showed an improvement of 12.7 and

[10] Available at http://members.unine.ch/jacques.savoy/clef/hungarianST.txt

[11] Available at http://members.unine.ch/jacques.savoy/clef/bengaliST.txt

Table 4 Query-wise performance and statistical significance test for English

Method	Query-wise performance		RI	*p* value
	Better	Poorer		
LOVIN	59	41	0.18	0.0214
PORTER	**63**	**37**	**0.26**	0.0088
LINGUISTICA	55	45	0.1	0.0598
MORFESSOR	56	44	0.12	0.0107
4-GRAM	48	52	−0.04	0.2397
YASS	54	46	0.08	0.0185
GRAS	58	42	0.16	0.0109
HPS	56	44	0.12	0.0161
PROPOSED	**63**	**37**	**0.26**	**0.0075**

Bold values indicate statistically better performance

Table 6 Query-wise performance and statistical significance test for Marathi

Method	Query-wise Performance		RI	*p* value
	Better	Poorer		
RULE	35	27	0.11	0.4027
LINGUISTICA	57	13	0.59	2.83E−06
MORFESSOR	57	15	0.56	3.88E−08
4-GRAM	55	15	0.53	2.70E−06
YASS	48	21	0.36	5.04E−05
GRAS	58	16	0.56	1.08E−08
HPS	55	15	0.53	4.97E−06
PROPOSED	**61**	**9**	**0.69**	**2.63E−09**

Bold values indicate statistically better performance

12.5%, respectively, in MAP. YASS performed fairly well but is found to be nearly 4% inferior to the proposed method. MORFESSOR and 4-GRAM performed almost equally and showed an improvement of nearly 8% in MAP. Rule-based Bengali stemmer and LINGUISTICA showed improvements of 6–7% in MAP as compared to no stemming.

The query-by-query analysis presented in Table 10 shows that the proposed method achieved the highest Robustness Index by improving 74 queries out of 125. We can infer from the *p* values (Table 10) that the retrieval results of the rule-based stemmer and LINGUISTICA are statistically insignificant as compared to no stemming. The proposed method showed better statistical performance among other baseline stemmers studied.

Jaro-Winkler Distance vs. Other Distance Measures

The proposed method clusters morphologically related words on the basis of the lexical distance between the word tokens sharing a common prefix of at least three characters. The choice of the distance measure is crucial as the quality of clusters depends on the distance measure being used. Therefore, in this subsection, we compared the retrieval performance of our proposed method based on the *Jaro-Winkler* distance with equivalence classes formed using other distance measures of historical importance, namely, *Hamming* and *Jaccard* string distance. The Hamming distance between two strings is the number of positions at which the corresponding characters in the two strings are different. It counts the number of substitutions required to convert one string into another. The Jaccard distance, on the other hand, is calculated by listing the unique character n-grams in two strings and comparing the ones they have in common. The Jaccard distance between two strings x and y is given by $1-\frac{|X\cap Y|}{|X\cup Y|}$, where X and Y denote the set of unique n-grams in strings x and y, respectively. We computed the Jaccard distance by comparing unique character bi-grams occurring in the strings. Besides these two distance measures, we also compared retrieval performance of our proposed method with the *Jaro* string distance to analyze the effect of the Winkler factor to the Jaro distance. The retrieval results of stemming based on word clustering using various distance measures are presented in Table 11. The best retrieval

Table 5 Retrieval results for Marathi

Total relevant documents: 1550 number of queries: 75 (FIRE 26–100)

Method	MAP	R-P	P@10	Rel. Ret.
NO STEM	0.2836	0.2938	0.3319	1307
RULE	0.2925 (3.14)	0.2987 (1.67)	0.3306 (−0.39)	1314 (0.54)
LINGUISTICA	0.3780 (33.29)	0.3743 (27.40)	0.3875 (16.75)	1441 (10.25)
MORFESSOR	0.4009 (41.36)	0.3936 (33.97)	0.3958 (19.25)	1420 (8.64)
4-GRAM	0.3875 (36.64)	0.3814 (29.82)	0.3889 (17.17)	1454 (11.25)
YASS	0.3456 (21.86)	0.3611 (22.91)	0.3625 (9.22)	1382 (5.74)
GRAS	0.4021 (41.78)	0.4042 (37.58)	**0.4194 (26.36)**	1449 (10.86)
HPS	0.3673 (29.51)	0.3764 (28.11)	0.3889 (17.17)	1400 (7.12)
PROPOSED	**0.4184 (47.53)**	**0.4157 (41.49)**	**0.4194 (26.36)**	**1459 (11.63)**

The best retrieval performance is marked in bold

Table 7 Retrieval results for Hungarian

Total relevant documents: 3158 number of queries: 150				
Method	MAP	R-P	P@10	Rel. Ret.
NO STEM	0.2159	0.2312	0.2728	1963
RULE	0.3147 (45.76)	0.3326 (43.85)	0.3654 (33.94)	2516 (28.17)
LINGUISTICA	0.2681 (24.18)	0.2837 (22.7)	0.3154 (15.62)	2493 (27.00)
MORFESSOR	0.3115 (44.27)	0.3310 (43.17)	0.3673 (34.64)	2580 (31.43)
4-GRAM	0.2736 (26.72)	0.2941 (27.21)	0.3265 (19.68)	2544 (29.60)
YASS	0.3149 (45.85)	0.3274 (41.61)	0.3716 (36.22)	2587 (31.79)
GRAS	**0.3245 (50.30)**	0.3397 (46.93)	0.3692 (35.34)	**2689 (36.98)**
HPS	0.3092 (43.21)	0.3334 (44.20)	0.3673 (34.64)	2637 (34.34)
PROPOSED	0.3229 (49.56)	**0.3398 (46.97)**	**0.3720 (36.36)**	**2689 (36.98)**

The best retrieval performance is marked in bold

performance of distance measures among various thresholds has been reported. The retrieval experiments in all the languages have been performed using the document collection and query sets reported in Table 1.

In Table 11, the best retrieval performance for all the languages is marked in bold, indicating that the equivalence classes formed using the Jaro-Winkler distance always happen to be the best. Among the other three baseline distance measures, Jaro gave the best retrieval performance, but the MAP values of Jaro are found to be nearly 3% inferior to the proposed method for English and Bengali and nearly 10% inferior to the proposed method for Marathi and Hungarian. The classes formed using Jaccard distance gave moderate results as compared to the unstemmed words in all the languages, but the performance is found to be inferior to both Jaro and Jaro-Winkler distance. Hamming distance performed poorest among all the distance measures and showed a very small improvement in MAP as compared to the unstemmed run in the various languages (except Bengali where the MAP values are reduced by 1.65% as compared to no stemming).

Table 8 Query-wise performance and statistical significance test for Hungarian

Method	Query-wise Performance		RI	*p* value
	Better	Poorer		
RULE	98	50	0.32	4.34E−08
LINGUISTICA	82	64	0.12	7.78E−05
MORFESSOR	99	54	0.30	5.17E−08
4-GRAM	86	62	0.16	1.21E−06
YASS	100	45	0.37	3.37E−08
GRAS	105	42	0.42	**1.02E−09**
HPS	94	55	0.26	6.67E−07
PROPOSED	**107**	**42**	**0.44**	5.24E−09

Bold values indicate statistically better performance

Analysis

In this section, we analyze the performance of various stemmers on the basis of the experimental results presented in the previous subsection. Figures 2 and 3 compare the mean average precision and number of relevant documents retrieved respectively by various stemmers across all the languages. The error bars in Fig. 2 represent standard error computed from the average precision values across the multiple queries. It is noticeable from the figures that the effect of stemming is found to be much more effective in Marathi and Hungarian as compared to English and Bengali. It is in accordance with the view that the efficiency of the stemmer increases with increase in the morphological complexity of the language.

Among the rule-based stemmers under study, the Marathi stemmer performed worse and provided marginal improvements in MAP and number of relevant documents retrieved. The Bengali rule-based stemmer performed fairly well, but the results are not found to be statistically significant as compared to no stemming. The reason for the low performance of Asian rule-based stemmers is that these stemmers perform light stemming and remove variants in nouns and adjectives. They do not consider variations in verb forms and commonly occurring derivational suffixes. The rule-based stemmers for English and Hungarian performed at par or better than the other baseline stemmers.

The language-independent MDL-based morphological analyzer, Linguistica, performed relatively worse in English and Bengali and gave moderate results in Marathi and Hungarian. The major limitation of Linguistica is that it is computationally intensive and takes relatively longer time to construct for moderate-size document collections. The recursive MDL-based Morfessor baseline method performed better in highly inflectional languages, Marathi and Hungarian, as compared to English and Bengali. In almost all the languages under analysis, the 4-gram approach performed inferior to the other corpus-based stemming methods except Linguistica. Another

Table 9 Retrieval results for Bengali

Total relevant documents: 3507 total queries: 125 (FIRE26–150)				
Method	MAP	R-P	P@10	Rel. Ret.
NO STEM	0.2904	0.2971	0.3352	2263
RULE	0.3109 (7.06)	0.3112 (4.74)	0.3352 (0.0)	2348 (3.76)
LINGUISTICA	0.3079 (6.03)	0.3030 (1.98)	0.3364 (0.36)	2363 (4.42)
MORFESSOR	0.3145 (8.30)	0.3034 (2.12)	0.3364 (0.36)	**2377 (5.04)**
4-GRAM	0.3139 (8.09)	0.3110 (4.68)	0.3366 (0.42)	2328 (2.87)
YASS	0.3188 (9.78)	0.3135 (5.52)	0.3372 (0.60)	**2377 (5.04)**
GRAS	0.3267 (12.50)	0.3187 (7.27)	0.3472 (3.58)	2361 (4.33)
HPS	0.3275 (12.78)	0.3240 (9.05)	0.3495 (4.27)	2344 (3.58)
PROPOSED	**0.3314 (14.12)**	**0.3278 (10.33)**	**0.3501 (4..44)**	2368 (4.64)

The best retrieval performance is marked in bold

language-independent stemmer, YASS, gave an average performance in almost all the languages. The string distance proposed in YASS is not suitable for agglutinative languages where long suffixes are present. The similarity measure used in our proposed method is suitable for all kinds of suffixing languages. It awards longest common prefixes between two-word pairs and considers insertions, deletions, and substitutions of characters while comparing the words. Graph-Based Stemmer (GRAS) and High Precision Stemmer (HPS) performed consistently well in terms of both mean average precision and relevant documents retrieved. However, GRAS seems to overstem words as it is highly dependent on recall rate.

Inflection Removal Experiments

In order to have a deeper insight into the quality of the stemmer, we decided to evaluate the performance of the stemmer directly without the effect of any specific application. Our test measures the ability of a stemmer to remove inflections from the variant word forms by comparing the set of words having the same stem (output of the stemmer) with the gold standards, i.e., readily available set of words having the same lemma (manual morphological annotations with lemmas). This test actually measures the capability of the stemmer to match the lemma. The results of this test can be used to decide whether a stemmer can replace a lemmatizer or not in applications like machine translation and POS tagging where lemmatizers work well. The results, however, are not very useful for applications where aggressive stemming is required. But we believe that it is not possible to construct a universal test for measuring stemmer performance as stemming is always dependent on the task to some extent.

Table 10 Query-wise performance and statistical significance test for Bengali

Method	Query-wise Performance		RI	*p* value
	Better	Poorer		
RULE	63	51	0.09	0.0762
LINGUISTICA	64	49	0.12	0.0771
MORFESSOR	57	50	0.06	0.0414
4-GRAM	58	53	0.04	0.0412
YASS	61	52	0.07	0.0198
GRAS	67	46	0.17	0.0072
HPS	67	47	0.16	0.0045
PROPOSED	**74**	**41**	**0.26**	**0.0005**

Bold values indicate statistically better performance

The inflection removal experiments are performed for English and Hungarian only due to lack of morphologically annotated data for Marathi and Bengali. Each stemmer has been trained on the Wall Street Journal document collection of English and Magyar Hirlap document collection of Hungarian (see "Description of Corpora" section). The English test corpus contains manually annotated texts of the Open American National Corpus (OANC)[12] containing 107,092 unique words. The Hungarian test corpus contains manually annotated documents from the Szeged corpus[13] containing 154,240 unique words. The inflection removal results of both English and Hungarian are presented in Table 12. We reported Precision (P), Recall (R), and *F* score (F) which are computed as

$$\mathrm{P} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} R = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} F = \frac{2\mathrm{PR}}{P+R} \tag{6}$$

where TP is the number of times the word in the equivalence class of stemmer matches a word in the lemma group (given by manually lemmatized data), FP is the number of times the equivalence class of the stemmer contains the wrong word, and FN is the number of times the equivalence class of stemmer missed the correct word. The *F* score measure considers both precision and recall

[12] Available at http://www.anc.org/data/oanc/
[13] Available at http://www.inf.u-szeged.hu/

Table 11 Retrieval results of stemming based on different string distance measures

English results				
Method	MAP	R-P	P@10	Rel. Ret.
NO STEM	0.2931	0.3286	0.4970	5956
HAMMING	0.2933 (0.07)	0.3288 (0.06)	0.5050 (1.61)	5948 (−0.13)
JACCARD	0.3040 (3.72)	0.3370 (2.56)	0.5010 (0.80)	6245 (4.85)
JARO	0.3139 (7.10)	0.3419 (4.05)	0.5060 (1.81)	6330 (6.28)
PROPOSED	**0.3236 (10.41)**	**0.3503 (6.60)**	**0.5120 (3.02)**	**6356 (6.72)**
Marathi results				
NO STEM	0.2836	0.2938	0.3319	1307
HAMMING	0.2873 (1.30)	0.2832 (−3.61)	0.3333(0.42)	1324 (1.30)
JACCARD	0.3759 (32.54)	0.3685 ((25.42)	0.3750 (12.98)	1425 (9.03)
JARO	0.3901 (37.51)	0.3899 (32.71)	0.3944 (18.83)	1448 (10.79)
PROPOSED	**0.4184 (47.53)**	**0.4157 (41.49)**	**0.4194 (26.36)**	**1459 (11.63)**
Hungarian results				
NO STEM	0.2159	0.2312	0.2728	1963
HAMMING	0.2187 (1.30)	0.2312 (0.0)	0.2745 (0.62)	2001 (1.94)
JACCARD	0.2504 (15.98)	0.2732 (18.10)	0.3048 (11.73)	2467 (25.67)
JARO	0.3002 (39.04)	0.3274 (41.61)	0.3650 (33.80)	2612 (33.06)
PROPOSED	**0.3229 (49.56)**	**0.3398 (46.97)**	**0.3720 (36.36)**	**2689 (36.98)**
Bengali results				
NO STEM	0.2904	0.2971	0.3352	2263
HAMMING	0.2856 (−1.65)	0.2848 (−4.14)	0.3136 (−6.44)	2260 (−0.13)
JACCARD	0.3095 (6.58)	0.3096 (4.20)	0.3280 (−2.15)	2348 (3.76)
JARO	0.3223 (10.98)	0.3252 (9.45)	0.3478 (3.76)	2351 (3.89)
PROPOSED	**0.3314 (14.12)**	**0.3278 (10.33)**	**0.3501 (4.44)**	**2368 (4.64)**

The best retrieval performance for all the languages is marked in bold

values, and thus, it is used as a final metric to compare the performance of the stemmer in inflection removal experiments.

The best values of Precision, Recall, and *F* score are boldfaced in Table 12. In the case of English, the Porter Stemmer and the proposed method gave the best results and yielded an *F* score of nearly 70%. GRAS, HPS, YASS, and LOVIN stemmers performed almost equally. The results of MORFESSOR are found to be similar to those of LINGUISTICA.

In the case of Hungarian, also the proposed method and the Snowball rule-based stemmer gave the best

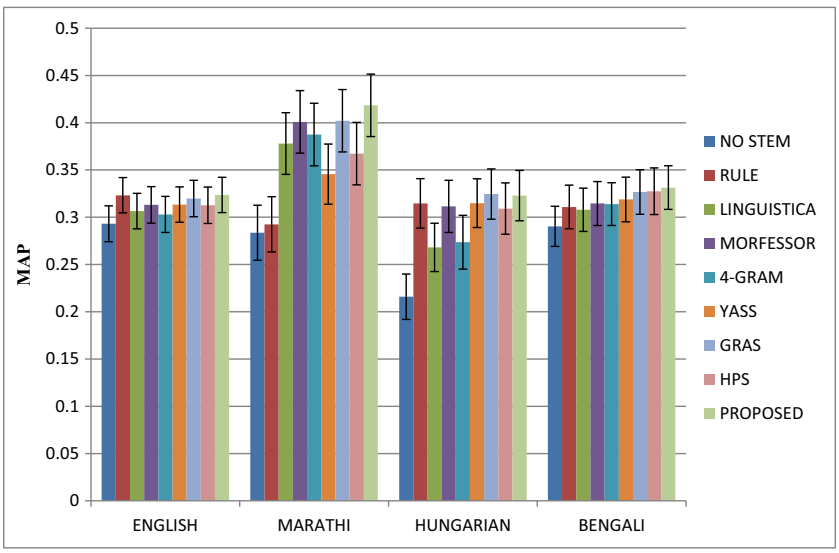


Fig. 2 Comparison of Mean Average Precision. The *error bars* represent standard error computed from the average precision values across the multiple queries

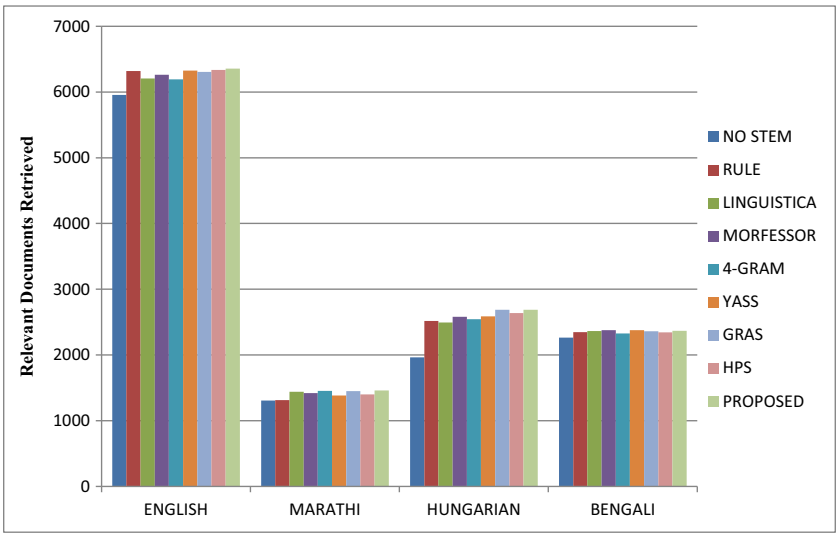


Fig. 3 Comparison of number of relevant documents retrieved

results and achieved an F score of 65%. GRAS also performed well and achieved an F score of 61%. MORFESSSOR and HPS also did fairly well in this test scenario. LINGUISTICA performed worst among all the stemmers in the case of Hungarian inflection removal experiments.

Conclusion

In this article, we presented an efficient fully automatic stemming method which clusters morphologically related words on the basis of lexical knowledge between the words in the corpus. The proposed algorithm does not require any language expertise or language-specific rules and is entirely based on the corpus. We tested the performance of our stemmer with the strong state-of-the-art corpus-based and linguistic stemmers in two different scenarios.

In the first set of experiments, the performance was investigated in an Information Retrieval task. The algorithm performs significantly better than all the baseline methods in retrieval experiments on four standard test collections from CLEF, FIRE, and TREC. In highly inflectional languages, like Hungarian and Marathi, the improvement in retrieval performance is found to be nearly 50% as compared to the unstemmed words. The query-by-query analysis also reveals that the proposed method improved the precision of a maximum number of queries in all the languages.

In the second set of experiments, we investigated the performance of the stemmer directly without any target application by comparing the outputs with manually annotated data. The proposed method performed significantly better than all the baseline corpus-based stemmers in these experiments as well.

The proposed unsupervised language-independent stemming method can thus be used as a multipurpose tool for various tasks such as the approximation of lemmas, improving retrieval performance or other Natural Language Processing applications.

Table 12 Inflection removal results

English results			
Method	Precision (%)	Recall (%)	F score (%)
LOVIN	60.2	67.4	63.5
PORTER	64.1	**77.3**	**70.1**
LINGUISTICA	53.3	61.8	57.2
MORFESSOR	52.5	61.5	56.6
YASS	57.9	63.7	60.7
GRAS	56.1	70.2	62.4
HPS	65.2	62.2	63.7
PROPOSED	**68.3**	71.1	69.7
Hungarian results			
RULE	68.4	**62.2**	65.2
LINGUISTICA	53.7	39.7	45.6
MORFESSOR	67.3	52.7	59.1
YASS	65.4	41.8	51.0
GRAS	68.2	55.2	61.1
HPS	62.3	51.3	56.3
PROPOSED	**69.8**	61.7	**65.5**

Compliance with Ethical Standards

Conflict of Interest The authors declare that they have no conflict of interest.

Ethical Approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Xu J, Croft WB. Corpus-based stemming using cooccurrence of word variants. ACM Trans Inf Syst. 1998;16(1):61–81.
2. Bhamidipati NL, Pal SK. Stemming via distribution-based word segregation for classification and retrieval. IEEE Trans Syst Man Cybern B Cybern: Publ IEEE Syst Man Cybern Soc. 2007;37(2): 350–60. http://www.ncbi.nlm.nih.gov/pubmed/17416163
3. Gupta V, Kaur N. A novel hybrid text summarization system for Punjabi text. Cogn Comput. 2016;8(2):261–77.
4. Toutanova K, Suzuki H, Ruopp A. Applying morphology generation models to machine translation. In Association for computational linguistics. 2008;pp. 514–522.
5. Shrivastava M, Bhattacharyya P. Hindi POS tagger using naive stemming: harnessing morphological information without extensive linguistic knowledge. In Proceedings of International Conference on NLP (ICON08). 2008.
6. Krovetz R. Viewing morphology as an inference process. In Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. 1993;pp. 191–202.
7. Dashtipour K, Poria S, Hussain A, Cambria E, Hawalah AYA, Gelbukh A, et al. Multilingual sentiment analysis: state of the art and independent comparison of techniques. Cogn Comput. 2016;8: 757–71. 1–15
8. Hu M, Liu B. Mining opinion features in customer reviews. AAAI. 2004;4:755–60.
9. Almeida TA, Silva TP, Santos I, Hidalgo JMG. Text normalization and semantic indexing to enhance instant messaging and SMS spam filtering. Knowledge-Based Systems. 2016.
10. Lovins JB. Development of a stemming algorithm. Mech Transl Comput Linguist. 1968;11:22–31.
11. Dawson JL. Suffix removal for word conflation. Bull Assoc Lit Linguist Comput. 1974;2(3):33–46.
12. Porter MF. An algorithm for suffix stripping. Prog Electron Libr Inf Syst. 1980;14(3):130–7.
13. Paice CD. Another stemmer. ACM SIGIR Forum. 1990;24(3):56–61.
14 Popovic M, Willett P. The effectiveness of stemming for natural-language access to Slovene textual data. J Am Soc Inf Sci. 1992;43:384–90.
15. Kraaij W, Pohlman R. Viewing stemming as recall enhancement. In Proceedings of the 19th annual International ACM SIGIR Conference on Research and Development in Information Retrieval. 1996 ;pp. 40–48.
16. Majumder P, Mitra M, Pal D. Bulgarian, Hungarian and Czech stemming using YASS. In Advances in multilingual and multimodal information retrieval. 2008;pp. 49–56.
17. Savoy J, Berger PY. Monolingual, bilingual, and GIRT information retrieval at CLEF-2005. In 6th Workshop of the Cross-Language Evaluation Forum, CLEF 2005. 2006;pp. 131–140.
18. Adam G, Asimakis K, Bouras C, Poulopoulos V. 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. 2010:pp. 389–397.
19. Dolamic L, Savoy J. Indexing and searching strategies for the Russian language. J Am Soc Inf Sci Technol. 2009a;60(12): 2540–7.
20. Dolamic L, Savoy J. Indexing and stemming approaches for the Czech language. Inf Process Manag. 2009b;45:714–20.
21. Paik JH, Pal D, Parui SK. 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). New York: ACM. 2011b; pp. 863–872.
22. Paik JH, Parui SK, Pal D, Robertson SE. Effective and robust query-based stemming. ACM Trans Inf Syst. 2013;31(4):1–29. doi:10.1145/2536736.2536738.
23. Orad D, Levow G, Cabezas C. CLEF experiments at Maryland: statistical stemming and back off translation. In: Proceedings of the Workshop of Cross-Language Evaluation Forum on Cross-Language Information Retrieval and Evaluation. Berlin: Springer-Verlag. 2001;pp. 176–187.
24. Goldsmith J. Unsupervised learning of the morphology of a natural language. J Comput Linguist. 2001;27(2):153–98.
25. Goldsmith J. An algorithm for the unsupervised learning of morphology. Nat Lang Eng. 2006;12(04):353–71.
26. Melucci M, Orio N. A novel method for stemmer generation based on hidden Markov models. In Proceedings of the twelfth International Conference on Information and Knowledge Management (CIKM'03). 2003;pp. 131–138.
27. Bacchin M, Ferro N, Melucci M. A probabilistic model for stemmer generation. Inf Process Manag. 2005;41(1):121–37.
28. Bacchin M, Ferro N, Melucci M. The effectiveness of a graph-based algorithm for stemming. In Digital libraries: people, knowledge, and technology. Springer; 2002. pp. 117–128.
29. Creutz M, Lagus K. Unsupervised models for morpheme segmentation and morphology learning. ACM Trans Speech Lang Process (TSLP). 2007;4(1):3. **article**
30. Creutz M, Lagus K. Unsupervised discovery of morphemes. In Proceedings of the ACL-02 workshop on Morphological and phonological learning. 2002; Vol. 6: pp. 21–30.
31. Creutz M. Unsupervised segmentation of words using prior distributions of morph length and frequency. In Proceedings of the 41st Annual Meeting of Association for Computational Linguistics. 2003;Vol. 1: pp. 280–287.
32. Creutz M, Lagus K. Induction of a simple morphology for highly-inflecting languages. In Proceedings of the 7th Meeting of the ACL Special Interest Group in Computational Phonology: Current Themes in Computational Phonology and Morphology. 2004:pp. 43–51.
33. Creutz M, Lagus K. 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). 2005; Vol. 1: pp. 51–59.
34. Kohonen O, Virpioja S, Klami M. Allomorfessor: towards unsupervised morpheme analysis. In Evaluating Systems for Multilingual and Multimodal Information Acces. Springer: 2008; pp. 975–982.
35. Majumder P, Mitra M, Parui SK, Kole G, Mitra P, Datta K. YASS: Yet Another Suffix Stripper. ACM Trans Inf Syst. 2007;25(4):18.
36. Jaro MA. Probabilistic linkage of large public health data files. Stat Med. 1995;14(5–7):491–8.
37. Winkler WE. String comparator metrics and enhanced decision rules in the Fellegi-Sunter model of record linkage. 1990.
38. Makin R, Pandey N, Pingali P, Varma V. Approximate string matching techniques for effective CLIR among Indian languages. In International Workshop on Fuzzy Logic and Applications. 2007;pp. 430–437.
39. Bilenko M, Mooney R, Cohen W, Ravikumar P, Fienberg S. Adaptive name matching in information integration. IEEE Intell Syst. 2003;18(5):16–23.
40. Christen P. A comparison of personal name matching: techniques and practical issues. In Sixth IEEE International Conference on Data Mining-Workshops (ICDMW'06). 2006;pp. 290–294.
41. Cohen W, Ravikumar P, Fienberg S. A comparison of string metrics for matching names and records. In KDD workshop on data cleaning and object consolidation. 2003 ;Vol. 3: pp. 73–78.
42. Paik J, Mitra M, Parui S, Jarvelin K. GRAS: an effective and efficient stemming algorithm for information retrieval. ACM Trans Inf Syst. 2011a;29(4):1–24.

43. Paik JH, Parui SK. A fast corpus-based stemmer. ACM Trans Asian Lang Inf Process. 2011;10(2):1–16. doi:10.1145/1967293.1967295.
44 Peng F, Lu Y. Context Sensitive Stemming for Web Search. Proceeding SIGIR '07 Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval. 2007;pp 639–46.
45. Brychcín T, Konopík M. HPS: high precision stemmer. Inf Process Manag. 2015;51(1):68–91.
46. McNamee P, Nicholas C, Mayfield J. Addressing morphological variation in alphabetic languages. In Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval. 2009; pp. 75–82.
47. Pirkola A, Keskustalo H, Leppänen E, Känsälä A-P, Järvelin K. Targeted s-gram matching: a novel n-gram matching technique for cross and monolingual word form variants. Inf Res. 2002;7(2):2–7.
48. Järvelin A. Applications of S-grams in natural language information retrieval. 2014.
49. Dolamic L, Savoy J. Comparative study of indexing and search strategies for the Hindi, Marathi, and Bengali languages. ACM Trans Asian Lang Inf Process. 2010;9(3):11.
50. Jaro MA. Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida. J Am Stat Assoc. 1989;84(406):414–20.
51. Brown PF, Desouza PV, Mercer RL, Pietra V, Della J, Lai JC. Class-based n-gram models of natural language. Comput Linguist. 1992;18(4):467–79.
52. Jain AK, Murty MN, Flynn PJ. Data clustering: a review. ACM Comput Surv. 1999;31(3):264–323.
53. Amati G, Van Rijsbergen CJ. Probabilistic models of information retrieval based on measuring the divergence from randomness. ACM Trans Inf Syst (TOIS). 2002;20(4):357–89.
54. Singh J, Gupta V. A systematic review of text stemming techniques. Artif Intell Rev. 2016:1–61. **article**. doi:10.1007/s10462-016-9498-2.
55. Sakai T, Manabe T, Koyama M. Flexible pseudo-relevance feedback via selective sampling. ACM Trans Asian Lang Inf Process (TALIP). 2005;4(2):111–35.