IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 18 October 2023, accepted 7 November 2023, date of publication 14 November 2023,
date of current version 1 December 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3332710

SURVEY

An Analytical Analysis of Text Stemming Methodologies in Information Retrieval and Natural Language Processing Systems

ABDUL JABBAR[1], SAJID IQBAL[2], MANZOOR ILAHI TAMIMY[1], AMJAD REHMAN[3], (Senior Member, IEEE), SAEED ALI BAHAJ[4,5], AND TANZILA SABA[3], (Senior Member, IEEE)

[1]Department of Computer Science, COMSATS University Islamabad (CUI), Main Campus, Tarlai Kalan, Islamabad 45550, Pakistan
[2]Department of Information Systems, College of Computer Science and Information Technology, King Faisal University, Al Hofuf 31982, Saudi Arabia
[3]Artificial Intelligence & Data Analytics Laboratory (AIDA), CCIS, Prince Sultan University, Riyadh 11586, Saudi Arabia
[4]MIS Department, College of Business Administration, Prince Sattam bin Abdulaziz University, Al-Kharj 11942, Saudi Arabia
[5]Department of Computer Engineering, College of Engineering and Petroleum, Hadhramout University, Mukalla, Hadhramout 50511, Yemen

Corresponding author: Saeed Ali Bahaj (saeedalibahaj@gmail.com)

This work was supported by the Deanship of Scientific Research, Vice Presidency for Graduate Studies and Scientific Research, King Faisal University, Saudi Arabia, under Project 5428.

ABSTRACT The exponential increase in textual unstructured digital data creates significant demand for advanced and smart stemming systems. As a preprocessing stage, stemming is applied in various research fields such as information retrieval (IR), domain vocabulary analysis, and feature reduction in many natural language processing (NLP). Text stemming (TS), an important step, can significantly improve performance in such systems. Text-stemming methods developed till now could be better in their results and can produce errors of different types leading to degraded performance of the applications in which these are used. This work presents a systematic study with an in-depth review of selected stemming works published from 1968 to 2023. The work presents a multidimensional review of studied stemming algorithms i.e., methodology, data source, performance, and evaluation methods. For this study, we have chosen different stemmers, which can be categorized as 1) linguistic knowledge-based, 2) statistical, 3) corpus-based, 4) context-sensitive, and 5) hybrid stemmers. The study shows that linguistic knowledge-based stemming techniques were widely used for highly inflected languages (such as Arabic, Hindi, and Urdu) and have reported higher accuracy than other techniques. We compare and analyze the performance of various state-of-the-art TS approaches, including their issues and challenges, which are summarized as research gaps. This work also analyzes different NLP applications utilizing stemming methods. At the end, we list the future work directions for interested researchers.

INDEX TERMS Text stemming, information retrieval (IR) systems, text classification, stemmer evaluation, technological development, natural language processing (NLP).

I. INTRODUCTION

The use of digital data and its online and offline processing has increased the size of textual data in multiple languages exponentially [1]. Consequently, a large volume of data is hosted on the web in various languages. This data is either organized or unorganized and requires efficient processing for extracting useful information and insights. The organizations are now building data lake houses that can contain both structured and unstructured data with data mining as a basic business requirement. Similarly, specialized applications like IR systems, data summarization applications, and question-answering-based chatbots are also candidate applications for efficient stemming methods [1]. Stemming is a process of

The associate editor coordinating the review of this manuscript and approving it for publication was Arianna Dulizia.

extracting the base form of the given word by removing its affixes. It is also recognized as a useful preprocess tool in many NLP applications [2]. Stemming is applied in a variety of applications such as information retrieval (IR), natural language processing (NLP), search engines, and domain vocabulary analysis as a preprocessing stage. Relevant processing and preprocessing tools are essential to process the textual content for these applications. At the start of NLP research, language processing tools were developed only for the English language however with time other languages received the focus of application developers and researchers, and tools for those languages also started developing. In an IR and NLP domain, the text pre-processing tools for analysis at the lexical, syntactic, morphological, and semantic levels are always required including text stemming. The researchers have proposed multiple TS algorithms for various national and international languages such as English, Arabic, Urdu, Chinese, German, French, and Italian. We surveyed papers from 1968 to 2023, but we selected the time frame from 2016 to 2022 to show the term ''stemming'' trend in Natural Language Processing (NLP). This is because the time frame from 2014 to 2022 covers the most recent and up-to-date research in the field. Google Trends data shows the increasing popularity of stemming in nlp as depicted in Fig. 1.

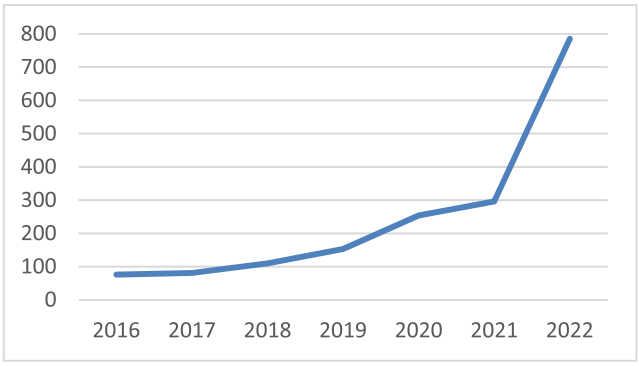


FIGURE 1. Interest in "stemming in nlp" since 2016 according to Google Trends (https://trends.google.com/trends).

Traditional methods for TS systems depend on linguistic knowledge, making the system language-dependent and language-specific with a high maintenance cost. On the other hand, the language-independent stemmers require a smaller-sized annotated dataset. Several issues associated with language-independent stemmers may include the selection of the appropriate threshold setting and performance dependence on selected parameters [2]. The context-sensitive approach is another choice to develop a stemmer [1] where stemming is performed based on query words.

Statistical stemming methods are language-independent, cost-effective, and easily adaptable; however, these approaches depend highly on annotated corpora. Large, annotated corpora are the initial and necessary steps of the statistical stemmers. Developing a stemmer for scarce resource languages is challenging [2].

Several state-of-the-art studies were conducted to investigate the diverse approaches of the TS systems. Most of them are focused on stemmers pertaining to certain languages or mainly dependent on analyzing the specific stemming approaches. Various survey studies [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15] did not consider performance influence factors, issues, and challenges to develop TS systems. Also, those studies were not discussed and analyzed using various TS evaluation parameters and mechanisms. In contrast to existing surveys, this paper highlights the status of past and recent advancements in TS approaches. The critical survey is presented in this article to help the research community in developing an effective solution to TS systems. For this reason, we have categorized the existing TS approaches into five categories: linguistic knowledge, statistically based, corpus-based, context-sensitive, and hybrid-based approaches. We analyze the merits and demerits of each approach in detail. In this review article, we have also identified several issues and challenges. We examined the factors that influence the performance of the TS process. So, these critical points must be considered when proposing a new TS solution.

A. CONTRIBUTION

Here is the list of main contributions of this research:

1. It provides a comprehensive overview of stemming techniques in a variety of language families. The detailed functionality of text-stemming methods is present and highlights their distinctive features. Performance analysis of the studies is provided as summarized tables to emphasize the features, benefits, and weaknesses of various TS techniques.
2. It analyzes and compares the performance evaluation metrics used in various languages' stemming methods. This comparison has considered the effects of dataset size, the nature of the data, and the choice of performance metrics. The merits and demerits of each evaluation metric are also discussed in detail.
3. It Identifies the issues and challenges faced in the design of text stemming algorithm including the factors that influence the performance of TS. We also point out various open questions related to language-independent stemming such as learning rules, learning semantic relations from the corpus, tuning statistical parameters, and using direct and indirect evaluation mechanisms.
4. Table 1 shows the research queries for research in the TS process.

The remaining paper is organized as follows. Section II describes TS systems and their usage as a preprocessing step in various NLP applications. A discussion of the related literature work is given in section III. In Section IV, the major text-stemming techniques are divided into four categories, and each category is critically analyzed and highlights the strengths and weaknesses of each text stemming technique. Section V portrays the performance comparison and analysis of text stemming evaluation mechanism. Section VI examines

TABLE 1. The summary of research questions.

S. No	Research questions
RQ1	What are the traditional stemming techniques used in the past?
RQ 2	Which kind of affixes are handled by the stemming techniques?
RQ 3	What are the applications of TS?
RQ 4	What is the main performance measure used for validating stemmers?
RQ 5	What are the main datasets being used by state-of-the-art stemming techniques?
RQ 6	What is the average size of data used to evaluate the stemmers?
RQ 7	What is the significance of using benchmark datasets?
RQ 8	What are the factors that affect the performance of a stemmer?
RQ 9	What are the TS issues and challenges?
RQ 10	Which parameters should be considered when designing a stemmer?
RQ 11	What are the limitations of present, stemming techniques?
RQ 12	What are the future research directions?

the various factors of text stemming that affect the performance of a stemmer. Section VII lists the various issues and challenges of TS and proposes the future research directions. Finally, concluding remarks are given in section VIII.

II. BACKGROUND

Text stemming is a morphological process to reduce a word to its standard form known as the root, by striping the attached affixes (prefixes, or/and suffixes, infixes) [2]. A root is a concrete word that is no longer divided into meaningful morphemes. For instance, in Fig. 2, we have presented various word forms of the root word ''consider''. A perfect stemmer must condense all such variant words of the root word. Usually, a stemmer may commit three types of errors: under-stemming, over-stemming, and mis-stemming [16]. When a part of the attached affix (prefix and/or suffix) is removed instead of a complete affix (prefix and/or suffix), that is known as an Under-stemming error. The over-stemming error occurs when some part of the query word is cut off along with the whole affix. The miss-stemming refers to the mistakenly chopping off the actual part of the input word.

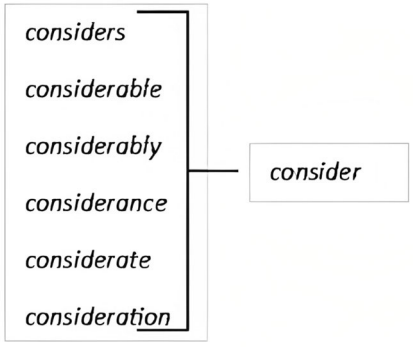


FIGURE 2. Example of variant word forms of ''consider''.

Many researchers utilized the TS algorithms as a pre-processing step in many texts/document analysis systems along with NLP systems to enhance the effectiveness of their systems. Here is the non-exhaustive list of TS uses in various applications.

A. INFORMATION RETRIEVAL (IR)

The TS aims to resolve the core issue of IR systems known as the vocabulary mismatch issue [17]. This issue arises when a query word does not match the various word forms found in the user query relevant documents. As an example, all the words shown in Fig. 3 would not be retrieved, because none of them match the user's query. Hence, the vocabulary mismatch problem is significantly reduced.

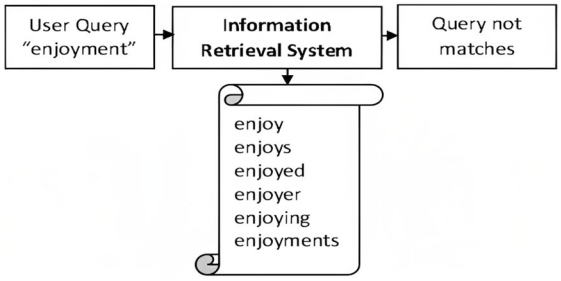


FIGURE 3. An example of vocabulary mismatch problem in IR systems.

Considering Fig. 3 example, the stemmer has reduced the query word ''enjoyment'' to ''enjoy''. In fact, all the words first are stemmed then are matched with query word ''enjoy''. In this way, all six words match with the user query. Reference [18] demonstrated that this TS process improved the effectiveness of IR systems.

B. MACHINE TRANSLATION SYSTEMS

An automated translation process that translates the text/speech from a source language to the target language forms the basis of machine translation (MT) systems. Studies like [19], used the stemming algorithm as a pre-processing step in English to the Indian language translation system. It proved the suffix separation can improve the performance of MT systems. Reference [20] indicated the use of stemmer to improve the performance of the Serbian to English MT system.

C. TEXT SUMMARIZATION SYSTEM

Text summarization systems read a large corpus of text and attempt to summarize the given text. As an example, the work in [21] validated the use of stemmer for the Arabic text summarization system resulting in improved performance.

D. SENTIMENT ANALYSIS SYSTEMS

These systems play an important role in analyzing social media text. Works like [2] have used TS, word removal, and tokenization as preprocessing tools in the sentiment analysis system [22]. Reference [23] Studied the sentiment

analysis systems and concluded that the tokenization, TS and stop words removal improve the performance of the system. Another work [24] evaluated the Arabic stemmer and concluded that the stemming helps improve sentiment analysis systems' performance.

E. NAMED ENTITY RECOGNITION (NER)

It is a challenging task for information retrieval systems. A preliminary step of the NER system involves tokenization, stop word removal, and stemming [25]. The study [26] shows that the stemming can reduce the vocabulary size, which eventually reduces the computation cost and increases the performance of the system.

F. QUESTION ANSWERING SYSTEMS (QAS)

Automatic recognition of questions and generating their answers is another challenging task. The authors in [27] demonstrated the use of morphological analyzers, such as stemmer, to increase the recall of the retrieval in QAS systems.

G. TEXT CLASSIFICATION AND CLUSTERING

Reference [28] experimentally verified that stemming is also helpful in text classification systems. Authors in [29] Evidenced the text-stemming process is used to enhance the effectiveness of the text classification system. Moreover, stemming improved the performance of clustering algorithms [30].

H. PART OF SPEECH (POS) TAGGING

In some languages, new words are concatenating morphemes like in agglutinative languages (Turkish, Hungarian, Korean, and Japanese). Agglutinative languages possess a multitude of morphological forms originating from a single word, leading to the emergence of the out-of-vocabulary (OOV) problem. This occurs due to the vast array of possible word variations within the language. To obtain the root of such words, the use of POS improves the performance of the NLP systems [31].

I. AUTOMATIC SPEECH RECOGNITION (ASR)

Many morphological changes are observed in speech data, increasing the vocabulary size exponentially and making the task challenging. In [32] researchers exhibited the positive influence of an ASR system and [33] confirmed that the stemming reduces the variant forms of words and improves the performance of an ASR system.

Stemming is used in a wide range of NLP applications such as text mining [9], word embedding [34], spell checkers [2], and tracing software engineering artifacts [35]. It is also used to extract various features and reduce the dimension of the feature vector to improve the system's performance. A comparative overview of the reviewed work is presented in Table 2.

TABLE 2. Summary of existing survey papers on TS.

Ref.	Research Questions (RQ)											
	1	2	3	4	5	6	7	8	9	10	11	12
[14]	✓	✓	✓	✓				✓	✓			✓
[2]	✓	✓	✓	✓	✓	✓		✓	✓			✓
[15]	✓	✓	✓	✓								
[3]	✓											
[4]	✓	✓	✓	✓					✓			
[5]	✓		✓	✓	✓				✓			✓
[6]	✓		✓	✓	✓				✓		✓	✓
[7]	✓		✓	✓							✓	
[9]	✓	✓	✓	✓	✓							
[10]	✓		✓	✓	✓	✓						
[11]	✓	✓										✓
[12]	✓		✓									
[13]	✓										✓	✓
Our article	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

A comprehensive evaluation and comparison of TS performance evaluation metrics is provided in [14]. In another work, Jabbar et al. [2] provided a detailed analysis of Urdu text-stemming algorithms. A comparative study is made by Kanan et al. [15] about the Arabic stemmer of Khoja, p-stemmer, and light10 stemmer. A multi-language (Arabic, Persian, and Urdu) stemming study is made by Jabbar et al. [3] to analyze different stemming methods and their applications in NLP.

1. Mustafa et al. [4] discuss the Arabic morphology and compare the Root-Based Stemmer, Light-Based Stemmer, Statistical Stemmer, Tagging Stemmer, and Artificial Intelligence based to Stemming approaches.

Here is a short list of selected studies that review the prominent stemming methods in different languages:

1. Singh and Gupta [5]: provide the systematic analysis of language-specific and independent stemming techniques.
2. Singh and Gupta [6]: The authors reviewed the applications of text stemming by comparing statistical and linguistic stemmers.
3. Dahab, Ibrahim, and Al-Mutawa [7]: examined Arabic stemmers, discussed their limitations, and demonstrated their usage in NLP applications.
4. Reference [9] studied the Persian stemmers and classified them into structural, statistical, and lookup table stemmers.
5. Moral et al. [10] mainly discussed the English stemmer and their usage in information retrieval.
6. Otair [11] compared the commonly used light stemmers in terms of affixes lists, and algorithms, and evaluated their performance in information retrieval applications.
7. Gupta and Lehal [12] surveyed common stemming techniques for Indian languages.
8. Anjali, Jivani, and Anjali [13] discussed the advantages and disadvantages of truncating, statistical, and corpus-based stammers.

III. RELATED WORK

Research literature shows that several surveys have been conducted to investigate TS techniques. This section briefly describes some recent approaches mainly focusing on Urdu. The authors in [2] have demonstrated a comparative analysis of Urdu text-stemming algorithms. Reference [3] presented a comprehensive review of multiple stemmers of Urdu, Persian, and Arabic languages. In this work, the authors described the advantages and drawbacks of every stemmer. Kanan et al. [15] compared the performance of the P-Stemmer, Light10 Stemmer, and Khoja Stemmer [84] on Arabic news articles. Mustafa et al. [4] reviewed the Arabic stemmer and discussed the challenges of the Arabic TS system. Singh and Vishal Gupta [5] surveyed various languages-dependent and language-independent stemmers, however mainly discussed unsupervised stemming methods. Singh and Vishal Gupta [6] reviewed the various stemming techniques included in the statistical and hybrid stemming techniques. The usage of stemmer in different NLP applications and issues about statistical approaches was also discussed. Many Arabic language stemmers are surveyed, and the comparative analysis was presented in [11]. Moghadam and Keyvanpour [9] reviewed only the Persian stemmer. Moral et al. [10] Mainly focused on English and European language stemmers. Gupta and Lehal [12] worked on Indian languages related stemmers. Jivani [13] surveyed the English stemmers and discussed the advantages and disadvantages of each. Merlini and Rossini [36] provide a comprehensive comparison of several classification and clustering techniques, namely the Naive Bayes Multinomial, Decision Tree, Support Vector Machines, k-Nearest Neighbors Classification, and the Random Forest algorithm. The decision tree model is implemented using the J48 algorithm, based on the C4.5 algorithm, and employed in conjunction with the WEKA platform. On the other hand, the Bayesian approach utilizes the NaiveBayesMultinomial classifier model for classification purposes. The k-nearest neighbors' technique is implemented using the IBk algorithm provided by WEKA. The training of a support vector classifier in WEKA is facilitated by implementing the SMO (Sequential Minimal Optimization) algorithm. WEKA provides an implementation of the Random Forest algorithm, which is a variant of the k-means clustering algorithm. This implementation in WEKA enables the utilization of Random Forest for clustering tasks, offering a powerful tool for analyzing and grouping data points based on their similarity. Atwan et al. [37] examined the impact of Arabic text stemming on three frequently employed classification algorithms: K-nearest neighbor, Naïve Bayes, and decision tree. They implemented these algorithms to evaluate their efficacy in handling Arabic text, both with and without incorporating a lightweight stemmer during the preprocessing stage. The authors addressing the limitations of commonly used classifiers such as K-nearest neighbor (KNN), Naïve Bayes (NB), and decision tree (DT) in the context of text classification (TC). One major weakness of these classifier algorithms is their limited performance when dealing with many features. Comparison of distinguishing characteristics of prior studies are depicted in Table 3.

TABLE 3. Comparison of features of prior studies.

Ref.	Characters
[14]	• Stemmers are divided into two categories: linguistic-based stemmer which includes rules base, template matching, and table lookup, and computational-based stemmer which is further divided into corps base and statistical. • Stemming errors such as under-stemming, miss stemming and over-stemming are analyzed. • The application of stemmers is elaborated. • Stemmer evaluation methods are divided into direct and indirect. • The pros and cons of each evaluation method are discussed. • Future directions are proposed.
[2]	• Only considered the Urdu language. • Urdu stemmers are classified into statistical stemmers and linguistic stemmers. • Narrated the Urdu morphology. • Text stemming pertaining to the Urdu language is discussed. • Comparative analysis of the state-of-the-art Urdu stemmer is examined. • Proposed the future direction.
[15]	• Only discuss the Arabic language stemmers. • present the comparative study of Arabic stemmer of khoja, p-stemmer, and light10 stemmer. • Recall, Precision, and F1-measure to evaluate. •
[3]	• Mainly considered the text stemming from Urdu, Persian, and Arabic. • Stemming errors such as under, miss, and over-stemming are analyzed. • Classified the stemmer into linguistic stemmer, corpus base stemmer, and hybrid stemmer. • The pros and cons of every stemmer are discussed.
	• Text stemming errors are described and analysis. • Recommended future work.
[4]	• Reviewed only Arabic stemmers. • Arabic stemming issues addressed discuss the Arabic morphology. • Classified the Arabic stemmers into Root-Based Stemmer, Light-Based Stemmer, Statistical Stemmer, Tagging Stemmer, and Artificial Intelligence based on Stemming approaches.
[5]	• Reviewed the stemmer of English language and other languages like Hungarian, Czech, Bengali, Marathi • Classified the stemmer into language-specific and language-independent stemming techniques. • The pros and cons of every stemming class are discussed. • Stemming applications defined. • Issues related to statistical stemming are described.
[6]	• Reviewed the stemmer of English language and other languages like French, Bulgarian, Bengali, Marathi • Text stemming is divided into rule-based, statistical, and hybrid stemmers. • The pros and cons of the stemmers are also discussed. • Direct and indirect evaluation mechanisms are described. • Stemming applications • Issue related to statistical stemmer.
[7]	• This survey studied only the Arabic stemmers. • A comparison of Arabic stemmer is presented in terms of usage in NLP and sensitivity for diacritics and context. • Stemming application mentioned. • Limitations of each stemmer are discussed.
[9]	• Mainly focused on the Persian language. • classified the Persian stemmers into structural, statistical, and lookup table stemmers. • The performance of the information retrieval system is measured by precision and recall. • Compare the size of Size of Test Collection

TABLE 3. *(Continued.)* **Comparison of features of prior studies.**

[10]	• mainly discus the English stemmer and their usage in information retrieval. • Stemming errors are divided into under-stemming, over-stemming, and mis-stemming. • Stemmers are evaluated in terms of strength based on committed errors. • Compare the dataset. • Present the comparison of data size to evaluate a stemmer.
[11]	• Overview of the Arabic morphology. • Divided the stemmer into root base stemmer and light stemmer. • To measure the strength of a stemmer a set of metrics is discussed. • Compares the most used light stemmers in terms of affixes lists, algorithms, main ideas, and information retrieval performance.
[12]	• Elucidate the English and Indian Languages like Gujarati, Marathi, Hindi and Bengali
[13]	• Mainly focused on English language stemmers. • Stemming errors are over-stemming and under-stemming. • Truncating, statistical, and mixed are Taxonomy. • elucidate the advantages and disadvantages of truncating, statistical, and corpus-based stammers. • Future directions are proposed
Our article	• A large variety of stemmers in various languages are described and analyzed. • The usage of stemmer in various NLP applications is deliberated. • Stemmers are divided into four categories: linguistic knowledge base, statistical base, corpus base, context base, and hybrid. • Identify the advantages and disadvantages of different types of stemmers. • Perform a comparative demonstration of different types of stemmers. • General factors which effect the performance of a stemmer are discussed. • Evaluation methods are divided into conflation base and application base evaluation bases such as information retrieval and classification. • A comparative analysis of various evaluation methods is presented. • The size of the data set that is used for evaluation is analyzed. • Issues and challenges of text stemming are analyzed. • Proposed the future directions.

IV. CLASSIFICATION OF TEXT STEMMING TECHNIQUES

TS techniques are classified into five main streams: linguistic knowledge-based methods, statistical-based, corpus-based, context-sensitive, and hybrid approaches. As the name implies linguistic knowledge-based methods use linguistic knowledge of relevant language for stemming. In statistical methods, various statistical techniques such as probability and N-gram frequency are used to build a stemmer. The corpus is used for corpus-based methods to determine the various features such as a lexicon similarity measure between words. In context-sensitive methods, first, the context of the query word is determined through context matrix/POS tagging then appropriate stemming rules are applied. Hybrid stemming approaches combine multiple techniques or methodologies from different categories to achieve more accurate and robust stemming results. These approaches may utilize a combination of linguistic knowledge-based, statistical, corpus-based, and context-sensitive methods to enhance the effectiveness of the stemming process. A visual example is shown in Fig. 4. These approaches are described in detail in the subsequent sections.

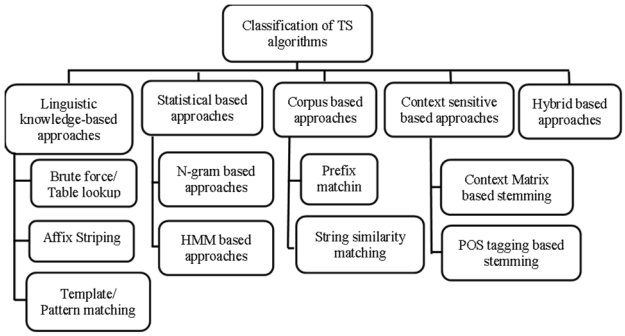


FIGURE 4. Taxonomy for state-of-the-art text stemming approaches.

A. LINGUISTIC KNOWLEDGE-BASED APPROACHES

In these approaches, the root is extracted using lexical and syntactic rules, and most of the stemmers are based on these methods [38]. Patel and Patel [19] built a linguistic knowledge-based stemmer for the Gujarati language. Alnaied et al. [39] design a linguistic rule-based stemmer for the Arabic IR system. Koirala and Aman Shakya [40] developed a rule-based stemmer for the Nepali language which functions in four steps. The first step was concerned with normalization and in 2^{nd} and 3^{rd} steps, the suffix is truncated using a suffix list. In the fourth step, the prefix is removed. Kassim et al. [41] and Kassim [41] designed two linguistic-based stemmers for texts in the Malay language. Azman et al. [42] constructed the tree structure, in which the center of the tree depicts the root verb and child nodes show its variant forms. Kaur and Preetpal Kaur Buttar [34] developed the 3,135 Punjabi root verbs and suffix removal rules. In this stemmer, firstly, the query word is searched in the Punjabi root words list and if found then it is returned as the root word. If the query word is not found in the Punjabi root verbs collection, then suffix removal rules are applied to obtain the stem. Atwan et al. [35] presented a linguistic knowledge-based stemmer for the Arabic language and proved that the proposed stemmer enhances the performance of the Arabic IR system. Bessou and Touahria [43] performed three sequences of steps to retrieve the stem of a given word. Normalization is applied to the query word as the first step and then different Arabic patterns are used from the look-up dictionary as the lexical analysis step. In the third step, an indexing phase is performed by the stem. Ali et al. [44] proposed an Urdu stemmer using a predefined prefix and a suffix exception list which is used to trim the prefixes and suffixes. The infixes are removed using infix rules. An example of an infix deletion rule is, if the Urdu words' length is five and it starts with ا [alif], then all ا [alif]

are removed such as اخبار [newspaper] stems to خبر [news]. Saeed et al. [45] proposed Reber stemmer for Kurdish Sorani text that operates in two steps. The first step deals with prefixes using predefined the prefix-list to remove the prefixes iteratively up to three letters' prefixes then the shorter string is resumed as a stem, as in Table 4 ''دمنگه کانیان'' is the shortest thus returned as a stem.

TABLE 4. Example of prefix handling rules.

Input of iteration	Iterations	Removed prefix	The output of each iteration
به کویدمنگه کانیان	First	ب	ه کویدمنگه کانیان
		به	کویدمنگه کانیان
کویدمنگه کانیان	Second	کو	یدمنگه کانیان
یدمنگه کانیان	Third	ی	دمنگه کانیان

In the second step specific suffixes are deleted in each iteration and from the obtained string after each iteration passed to the next iteration as an input, and the shortest string ''ده نگ'' is an acclaimed stem as depicted in Table 5.

TABLE 5. Example of suffix handling rule.

Iterations #	The input of each iteration	Removed suffix	The output of each iteration
First	ده نگه کانیان	ن	ده نگه کانیا
		ان	ده نگه کانی
		یان	ده نگه کان
Second	ده نگه کانی	ی	ده نگه کان
	ده نگه کان	ن	ده نگه کا
	ده نگه کان	ان	ده نگه ک
	ده نگه کان	ه کان	ده نگ
Third	ده نگه کان	ن	ده نگه کا
	ده نگه کان	ان	ده نگه ک
	ده نگه کان	ه کان	ده نگ
	ده نکه ک	ه ک	ده نگ
Fourth	ده نگه ک	ه ک	ده نگ

Suryani et al. [39] presented rules-based Sundanese Stemmer (RBSS) that starts with the left-side affix deletion in a sequence. This method first removes left-side affixes, then middle-side infix/ allomorph is removed, and finally, right-side suffixes are truncated to derive the stem as shown in Table 6.

Jabbar et al. [46] developed a few resources for Urdu text stemming. Ali et al. [47] developed the Urdu stemmer with a four-stage operation. Those four stages function in a linear fashion. In the first stage, the prefixes are stripped using predefined prefix removal rules. The second stage handles the infixes with a predefined infix handling template. The third stage removes the suffixes from the query word such as اداروں [Institutes] reduced to ادار and passed to the fourth stage, in which the letter ہ [hy] is appended at the end of ادار and the resulting stem is ادارہ [Institute]. Abainia et al. [48] designed an Arabic stemmer that retrieves the stem from a query word in six steps. The 1st, 3rd, and 6th steps perform stemming on Arabic verbs while steps 2, 4, and 5 trim the affix from the Arabic noun. To avoid under and over-stemming errors,

TABLE 6. Illustration of procedure to remove affixes.

Input string	Steps	Removed affixes	Output string
Dipangmaracaankeun [being read by someone else]	left-side affixes deletions	"di" [Prefix]	pangmaracaankeun [asking someone to read something]
		"pang-keun" [confix]	"maracaan" [some people are reading]
	middle-side infixe/ allomorph	"ar" [infix]	"macaan" [reading many times]
		"m" [allomorph]	"bacaan" [something to read]
	right-side suffixes	"-an" [sufix]	"baca" [to read] the stem

the authors have considered the word length along with the attached affix length to remove the affixes.

For instance, the Arabic word كتابهم [their book] in which هم [their] is a suffix that satisfies the condition. For example, if the length of a word is greater than two and the suffix length is two, then the suffix is removed. In the case of the word فهم [understanding] in which, the هم [their] suffix did not meet the condition. So, the word فهم [understanding] remained unchanged. The first step concerns normalization in which some Arabic characters are replaced with others as ى [Alif MaqSura] replaced with ي [yaa] and some common Arabic conjunction characters are removed. The second step determines the prefixes and removes them. The third step deals with suffixes, if a true suffix is found then the stem is derived using appropriate rules. The fourth step transforms the plural to singular by deleting the affixes (infixes/suffixes) such as اسراب [swarms] stem to سرب [swarm]. Step five transforms the feminine to the masculine form of query words. Finally, step six deletes the verb suffix such as سافعل [I will do] stem to فعل [de did]. Selvaramalakshmi et al.'s designed stemmer [49] first tokenize the query word as first five letters as Pgrams and the last seven characters considered Sgram. Then Pgrams are matched with a prefix-list iteratively, in each iteration a later is removed and the remaining letters are matched with a prefix-list. After this Sgram is also matched with a predefined suffix list, and truncates the suffix letter one by one until matched with the predefined suffix list as shown in Table 7 in which the stem of ''drinkable'' is retrieved when Sgram ''able'' is matched with the predefined suffix.

Kassim et al. [41] developed a stemmer for the Malay language that recognized infix using a pattern if the pattern did not match then it looked up in the stem-word dictionary and relevant stem retrieve. To remove the prefix, suffix, and confix, first match the query word with derivative dictionary entries, if found, then its corresponding stem is returned, otherwise affixation stemming rules are applied and extract the stem as in the example given in Table 8.

TABLE 7. Example of the tokenization process.

Query word	token	Description
drinkable	drink	Pgrams not found in prefix-list
	rink	
	ink	
	nk	
	k	
	inkable	Sgram
	nkable	
	kable	
	able	Sgram match
Produce stem drink		

TABLE 8. Sample of affix recognition and deletion rules.

Steps	Query word	Affix recognition and deletion	Obtained stem
1	Pembaca [reader]	prefixation	Baca [read]
2	makananmu [his/her food]	suffixation	makanan [food]
3	mempertemukannya [his/her findings]	confixation	temu [finding]
4	telunjuk [pointing fingers]	infixation	tunjuk [show]

Aldabbas et al. [50] presented an Arabic stemmer that constructs the regular expression based on the already defined prefix and suffix removal patterns. Finally, query and stem words are compared, and a stem is extracted if both have the same meaning. The affixes are eliminated by a regular expression, and then the dictionary's definition of both query word and stem in Microsoft word dictionary is checked, if the meaning of both is same, then it is extracted as a stem.

TABLE 9. Example of produce stem by CS arguments.

Set of words	Produced stem	Translation	Arguments
HIMARË	HIM	οι ύμνοι	
HIMNET	HIMN	των ύμνων	CS(HIM)
HIMNIN	HIM	τον ύμνο	
HIMNIT	HIM	του ύμνου	

TABLE 10. DS arguments case to obtain the stem.

Set of words	Produced stem	Translation	Arguments
KOSTA	KOST	όνομα ανθρώπου	DS
KOSTON	KOST	κοστίζει	

Karanikolas [51] presented a set of consecutive triplet word stems and the translation is provided as the stemmer. The stemmer removes the longest suffix using pre-construct suffix lists and the suggestion is obtained on the produced stem from the human expert, however, if the expert did not declare the decision, then the stemmer's result is accepted. The example of the produced stem along with arguments is mentioned in Table 9, in which the argument Common Stem (CS) means stemmer produces more than one stem from the set of the words though this set of the words has CS arguments. In Table 11 the stemmer gives a common stem, but it is a different stem as mentioned in the arguments column Different Stem (DS).

TABLE 11. Example of rules description with instances.

Affix type	Rule description	Examples

Prefix 'ت'	If the input word's length > 3 and the 2nd letter is "ر and the 3rd letter is 'ي' or 'و' then eliminate the 1st letter.
	Else
	delete the 1st and 2nd char.
	End if
		تراس [to feel] → راس [feel]

Suffix 'ن'	If query word's length > 3 and 2nd last character ="ا" and 3rd character is 'و'
	Then remove the last two characters.
	Else
	Remove the last character.\
	End if
		بنتوان [helping] → بنتو [help]

Sulaiman et al. [52] developed a TS algorithm for Malay text which comprises mainly two stages: in the first stage the affixes (prefixes and/or suffixes) are eliminated using predefined rules like the example given in Table 10 and the in second stage Spelling Error Detector Rules (SEDR) are applied to the prior extracted stem.

Kasthuri and Kumar [53] defined stemmer for the Tamil language that first removed the prefixes using prefix-list and record, such as எகாலம [which period] stem to, காலம [period] second it eliminates the suffixes by predefined list of suffixes, for instance, ட, றதவகள [birds] stem to பறதவ [bird]. Third, if the query word is an adjective and tense, then it is replaced equal verb. Fourth, if a stem is not found, a possible suffix is generated and added to the rules list. In step five steps one to four are repeated until the stem is found. Karaa, [54] addressed the following six deficiencies of the Porter stemmer [38] to build an English rule-based stemmer.

- Porter stemmer [38] does not handle the irregular forms of verbs such as "bought".
- Verbs ending with 's' are ignored, for instance, focus.
- When a word has a vowel letter, then Porter stemmer [38] replaces the ending 'y' with 'i' but does not conflate the English word 'try', 'tries', and ' tried'.
- Verbs ending in a double consonant make errors, for instance, "ebbed" is wrongly stemmed to "eb", the actual stem is 'ebb'
- It did not handle the present/ past participle derivations, for example, 'studiedly' stem to 'studiedli'
- Porter stemmer [38] has ignored several suffixes, for instance, "'est', 'ist', 'tary', 'tor', 'sor', 'sory', 'nor', 'ship','acy', 'ee"'.

The stemmer proposed by Karaa, [55] functions in five steps, and specific suffixes are handled in each step. The inflectional and derivational morphologies are handled in the first two steps. Further, suffixes are removed, and recoding rules are

applied to obtain stem in step 3 and step 4 respectively. Finally, the irregular form of a verb is reduced in the last step. Mishra and Prakash [56] defined the stemmer which combines the brute force and suffix removal approaches. The algorithm first checks the query word in the stem-word dictionary and returns its corresponding stem, if found. Otherwise, the stemmer moves to the second step, in which the suffix is stripped using a suffix list and extracts the stem. El-Beltagy and Ahmed Rafea [57] presented stemmer for the Arabic language. The researcher developed single letter affixes (prefix and suffix) and compound affixes (prefix and suffix) lists. First, the stemmer truncated the single letter prefix, then the compound prefix is removed. Lastly, the infixes are handled by predefined patterns. An example of such a transforming pattern is given in Table 12.

TABLE 12. Rules description with example words.

Rule id	Conditions	Rules	Examples
R1	The query word's length is five letters 1st and 4th letters are ا [alif] and the second character is not ي	Remove the 1st and 4th letter ا [alif]	اشجار
R2	The query word's length is four characters, and the 1st letter is an ا [alif]	purge the 1st letter ا [alif]	أشهر
R3	The query word's length is five and 3rd character is a ا [alif]	censor the 3rd letter ا [alif]	مراكب

Savoy [58] developed aggressive and light stemmers for the Hungarian language, using the affix striping technique, and evaluated them on the CLEF document collection, the results show that the aggressive stemmer performs better than the light stemmer. The performance of Lovin et al. [59] stemmer is promising due to the non-iterative nature of the algorithm. However, it is complicated because of the large suffixes list i.e., 1200 suffixes. Porter [38] proposed a stemmer that removes all the suffixes iteratively. Jabbar et al. [60] devised a set of rules to handle infixes, reducing Mohmil words, and incorporating multi-level suffixes. Additionally, they constructed rules for identifying affixes and deriving the stem using prefixes and suffixes rules. By evaluating the proposed approach against the state-of-the-art stemmer, the authors achieved an impressive accuracy of 94.92%.

B. STATISTICAL-BASED APPROACHES

In statistical approaches, the linguistic knowledge of the underlying language as well as knowledge about training data sets are not required. In the N-gram method, a continuous word is divided into different N-characters, and a frequently occurring sub-word is extracted as a stem. In the HMM-based approach, a word is constructed using Finite Automata, in which some probability function determines the output of each transition. Usually, some initial state represents the stem and the final states denote the suffix.

Sadia et al. [61] used an N-gram-based technique and tested on Bangla language. Pande et al. [62] also used an N-gram technique to develop a stemmer and frequency of the N-gram to determine the stem's possibility. Dadashkarimi et al. [63] proposed a statistical stemmer to extract the root from the inflectional and derivational forms of the word. The method learns the linguistic pattern from the huge collection of web pages using Minimum Edit Distance (MED) algorithms and Parts of Speech (POS) tagger. Every rule assigned a score using a profanity function in Eq. 1.

$$P(R) = P\left(W'/W\right) \tag{1}$$

where, $P(R)$ refer the probability of pair of words W' and W.

Brychcín and Konopík [64] presented another statistical stemmer, which works in two stages. In the first stage, different word clusters of datasets were designed and the training data from various clusters were constructed based on the longest matching prefix. Then, various features such as suffix and N-gram probability, length of the word, and global statistics were extracted from the prior created clusters. The maximum entropy was classified using these features to take out the stem. Ferilli et al. [65] used the Kronecker function for similarity measure that produces one in case of an equal string, otherwise zero, and grouping the words based on high and low prefix likeness. The threshold value is used to disjoint two groups by computing the average of all similarities. Finally, the longest subsequence common to all its elements is returned as a stem. Pande et al. [66] used 4-gram as an initial prediction for the stem. The given word is tokenized 4-gram, 5-gram, 6-gram up to word length. Having a minimum frequency N-gram is returned as a stem. Husain, [67] proposed a stemmer that splits the given words into N-grams as a pair of (stem | suffix) and generates the possible stem and suffix list as given below:

$$word = \{(stem_1 \mid suffix_1), (stem_2 \mid suffix_2), \ldots (stem_n \mid suffi_n)\}$$

For a word "ages", the equation could be written as:

$$ages = (NULL|ages), (a|ges), (ag|es), (age|s), (ages|NULL)\}.$$

Finally, the generated suffix is clipped on the highest frequency and using the longest string. Pandey and Siddiqui [68] used the "split all" method in the Hindi language. Ahmed and Andreas Nürnberger [69] used the N-gram technique to tokenize the query word into the possible stem, then measure the similarity by "edit distance" measure, to group related words.

Al-Shalabi [70] developed a successor variety of Arabic stemmer that determines the stem and affix (prefix and/or suffix) boundary. The threshold and entropy values are used to segment the query word into the stem and affix (prefix and/or suffix). The "cut off" method with a suitable threshold value gave 80 % accuracy which is better than the entropy method which achieved 75% correctness. Bacchin et al. [71] proposed

a probabilistic model to retrieve the stem, it generated all the possible substrings as prefixes and suffixes from the query words then the highest probability prefix was chosen as a stem. It operates in two steps, the first is global and the other is local. In the global step, some basic linguistic knowledge is inferred from the dataset using Eq. 2 and Eq. 3. Splitting corresponds to a stem, and a derivation is determined by Eq. 4.

$$Pr\,(x_i) = \sum_{j=1}^{N} Pr\,(x_i|y_i Pr\,(y_j) \quad i = 1, \ldots, N \tag{2}$$

$$Pr\,(y_i) = \sum_{i=1}^{N} Pr\,(y_j|x_i Pr\,(x_i) \quad j = 1, \ldots, N \tag{3}$$

In which x_i and y_j are prefix and suffix respectively with index i and j. Equation 4 is used to find the most probable split ω^* of z

$$\omega^* = \arg\max Pr\,(\omega) \quad \omega \epsilon \Omega\,(z) \tag{4}$$

Mayfield and McNamee [72] presented a statistical base English stemmer using single N-gram methods and Melucci and Orio [71] presented a statistical base stemmer using Hidden Markov Models (HMM).

C. CORPUS-BASED APPROACHES

These approaches used mathematical computations such as Co-occurrence, Maximum Entropy Method (MEM), and different graph-based techniques to expose latent similarities between words in the training corpus. In this method, the different distance measuring techniques are applied, and then built various classes using clustering techniques. Finally, the centroid of classes is determined. Several researchers utilized the corpus-based approaches to develop a stemmer, which is detailed described below paragraph.

Singh and Alotaibi [73] divided the words among clusters using the lexical and co-occurrence similarity along with the Potential suffix pair frequency. Lexical similarity is measured by Eq. 5 and co-occurrence is calculated by Eq. 6.

$$lexical_{similarity(w_1,w_2)} = \frac{p}{n} \sum_{i=1}^{n'} (0.5)^i \times s_i \tag{5}$$

where p is the length of the common prefix, n is the number of letters in w_1 and w_2, the smaller string pad to null.

s_i compare the letters in both strings if both have identical letters then the $s_i = 1$ otherwise 0.

$$co - occurence_{similarity} = \frac{2 \times df\,(w_1, w_2)}{df\,(w_1) + df\,(w_2)} \tag{6}$$

where, $df\,(w_1, w_2)$ is the co-occurrence frequency of words.

$df\,(w_1)$ frequency of w_1 and $df\,(w_2)$ frequency of w_2.

Alotaibi and Vishal Gupta [73] proposed a three-step language-independent stemmer, the first step is concerned with measuring lexical and structural distance by Jaccard distance (see Eq. 7) between unique Uni-gram words and finding the least distance between pairs of objects by Eq. 8. In step two a complete linkage algorithm is used to group words whose distance is computed in the prior step, finally, in step three, the importance of each variant of a query term is computed by Eq. 9.

$$D\,(w_1 - w_2) = 1 - \frac{|X \cap Y|}{|X \cup Y|} \tag{7}$$

Here, X and Y represent the sets of unique Unigrams of the words w1 and w2 respectively.

$$D\,(C_1, C_2) = max_{x \in c_1, y \in c_2} D\,(x, y) \tag{8}$$

$D\,(C_1, C_2)$ refer to the maximum distance between cluster C_1 and C_2

The weight of the variability of a query term as

$$\mathrm{W\,(t.q)} = \sum_{\mathrm{i=1}}^{\mathrm{n}} \frac{\mathrm{df\,(q_i)}}{\mathrm{N}} \mathrm{(PMI\,(t,q_i)\,,\,0)} \tag{9}$$

In which (see eq. 10) Pointwise Mutual Information (PMI) scores, refer to the co-occurrence of two words in the corpus.

$$PMI\,(t, q_i) = log_2 \frac{P\,(t, q_i)}{P\,(t) P(q_i)} \tag{10}$$

W (t.q) showed the weighted sum of the PMI scores between a target word t and a set of query words q.

Singh and Vishal Gupta published another method of stemming [74] which functions in two phases: In the first phase, various classes are constructed based on common prefixes of three letters and determine the distance between two strings using Jaro-Winkler distance (Eq.11). In second phase the method minimizes the prior calculated distance to threshold value using Eq.12 and average linkage is used to group the morphologically similar words.

$$Jaro - Sim\,(w_1, w_2) = \Phi = \left(\frac{\boldsymbol{c}}{\boldsymbol{l_1}} + \frac{\boldsymbol{c}}{\boldsymbol{l_2}} + \frac{\boldsymbol{c} - \boldsymbol{t}}{\boldsymbol{c}}\right) \tag{11}$$

where matching letters should not be farther than $\left\lfloor \frac{max(\boldsymbol{l_1}, \boldsymbol{l_2})}{2} \right\rfloor - 1$.

Where,

c refers the number of matching letters in two words.

l_1 is the length of the first word.

l_2 is the length of the second word.

t refers the number of matching letters that appear in different orders. Then, divide this number by two.

$$D_{avg}\left(C_i, C_j\right) = \frac{1}{|C_i| \times |C_j|} \sum_{x \epsilon C_i, y \epsilon C_j} d\,(x, y) \tag{12}$$

where:

D = distance between clusters Ci and Cj

Ci = the cluster with index i

Cj = the cluster with index j

$|\,Ci\,|$ = length of cluster Ci

$|\,Cj\,|$ = length of cluster Cj

Boukhalfa et al. [75] developed a graph-based Arabic stemmer, in which each letter is represented in a connected graph along with the main characteristics of the letters. Paik, et al. [76] proposed a corpus-based stemmer that comprises three main steps: it first groups words based on similarity, and then using co-occurrence measure between

groups it constructs the prior step and finally query depends on implying the semantic relationship. Its mathematical formula is given in Eq. 13.

$$RC\,(a,b) = \frac{-\log\left(Prob\left(df\ (a,b) = m\right)\right)}{-\log(Prob\left(df\ (a,b) = MIN(n_1,n_1)\right))} \quad (13)$$

In which, n_1 and n_2 documents contain the words a and b respectively, whereas m documents contain both words. Paik & Paik and Parui [77] presented a corpus-based stemmer that functions in three steps: in the first step, k-equivalence classes are constructed by the potential suffix and common prefix. The potential suffix is the generated suffix from the corpus, which has a higher frequency than a certain cutoff threshold value and the common prefix becomes the root of the class. In step two the strengths of the prior developed classes are calculated as Eq. 14.

$$strength\,(R) = \frac{size\ of\ the\ potential - class\,(R)}{size\ of\ the\ generated - class\,(R)} \quad (14)$$

A common prefix-based root may have more than one valid root. In step three, the potential root is determined by the strength and then cut off threshold. Reference [78] presented a corpus-based stemmer that functions in two steps. In the first step, the stemmer produces the 3-gram of the query words to build clusters. In the second step, developed clusters are refined using Dice Distance (Eq. 15).

$$S_{ab} = \frac{2C_{ab}}{C_a + C_b} \quad (15)$$

where,

a and b are two words and S_{ab} is the Dice measure.

The words are clustered in large equal classes using the value of the Dice Similarity measure and the Complete Linkage Clustering Algorithm (CLA). Three different values of threshold have been practiced i.e., $t = 0.5, t = 0.6, t = 0.7$.

Then the co-occurrence of each word in a similar class is determined by the Expected Mutual Information Measure (EMIM) called EM (Eq. 16), in which a and b are two terms and $En\,(a,b)$ represents the expected number of co-occurrences of term a and b. Using the EM score the Optimal Partition Algorithm (OPA) is applied for clustering.

$$EM\ (a,b) = max\left(\frac{n_{ab} - En\,(a,b)}{n_a - n_b}, 0\right) \quad (16)$$

Zitouni et al. [78] also experimented with combining both the Dice measure and the Expected Mutual Information Measure (EMIM) similarity measure such as Eq. 17.

$$SEM = \frac{S_{ab} + EM\ (a,b)}{2} \quad (17)$$

Majumder et al. [79] calculate the string similarity using Eq. 18.

$$\mathrm{D_1\ (X,\ Y)} = \sum_{\mathrm{i=0}}^{\mathrm{N}} \frac{1}{2^{\mathrm{i}}} \mathrm{P_i} \quad (18)$$

where, $\mathrm{P_i}$ is 0 if there is a match in the *ith* position of X and Y. Otherwise $\mathrm{P_i}$ is 1.

Using the prior distances Complete-linkage clustering construct and center words returned as a stem word.

Korenius et al. [80] utilized four hierarchical clustering methods i.e. The single linkage, group average linkage, complete linkage, and Ward's clustering methods to develop stemmer for Finish language. The author proved that lemmatization performs better than stemmer in clustered Finish documents.

D. CONTEXT-SENSITIVE APPROACHES

Context is sensed before the query is sent to the search engine, which minimizes the unwanted query scope expansions. Accordingly, it reduces the time and space complexity. Maximizing the semantic similarity between the query word and the retrieved term improves the precision [81]. Bölücü and Can [31] presents a joint PoS tagging and stemming model that integrates two Hidden Markov Models. It is tested in English, Turkish, Finnish, Hungarian, and Basque languages. Basu et al. [82] stemmer functioned into two phases: in the first phase, using the string similarity technique various variant forms of query word are generated, then in phase two using word2vec approaches to retrieve the stem according to the context.

Mezzi et al. [83] proposed a Semantically Enriched Context-Aware Stemming (SECAS) algorithm that incorporates the dictionary and rule-based stemmer features. The first phase contains the Wordnet database that is used to produce the stem according to the Parts of Speech (POS), and the second phase is rule-based and is used for Context-Aware Stemming (CAS) [84] with additional preprocessing steps to produce the stem. After the results obtained from the prior phases, are compared and SECAS algorithm retrieves the most relevant stem. Bölücü and Burcu Can [31] used the Bayesian model along with Hidden Markov Models (HMMs) for POS tagging and stemming jointly for the Turkish language. Boukhari and Omri [85] proposed a context-sensitive English stemmer that compromises six steps: in step one, compound words related suffixes are removed, and steps two and three are related to extraction of the context of the query word. Various forms of the input word appear in a context and all forms are reduced to the same stem. For example, the words that appear in a context are 'chair', 'chairs', and 'chairman', stem to ''chair. Step four deletes the suffix using a suffix list. In step five the elimination of redundant letters is performed. Final, transforming rules were applied in step 6 to retrieve the stem.

Agbele et al. [84] proposed a Context-Aware Stemming (CAS) algorithm for the English language, which removes the suffixes iteratively, after every iteration addition or deletion, and/or substitution is performed to retrieve the semantically correct stem, for instance, the English word 'filing' stem to 'fil' after removing the 'ing' suffix, and the ending of the derived stem 'fil' contains consonant vowel consonant (CVC) pattern, so the 'e' is appended at the end of the derived stem. Adam et al. [86] presented stemmer which first tokenizes the

query text and tags the POS, then the suffix is removed using a predefined list of suffixes according to parts of speech. Al-Shammari and Lin [87] proposed a context-sensitive stemmer for the Arabic language. This method first recognizes the nouns and verbs from the input text and then applies the light stemmer [88] on the noun and Khoja and Garside [89] root-based stemmer on the verbs. Sarkar and Bandyopadhyay. Reference [90] presented a stemmer for the Bengali language, which performs the parts of speech tagging of the query word then according to suffix removal rules, suffixes are removed. Lovins [59] proposed a context-sensitive, longest-match stemming algorithm for the English language. It is the first popular and effective English stemmer mentioned in the literature. Lovins [59] stemmer has two steps, one is the basic stemming procedure in which the suffix is removed based on sixty rules. The second step is the recoding procedure in which different adjustments are performed on the obtained stem, to convert it into a valid word.

E. HYBRID APPROACHES

When more than one stemming method are combined for stemming purpose, the new approach is known as the hybrid method. Numerous hybrid algorithms have been proposed in the literature. Some of them are reviewed below:

The authors in [91] combined the dictionary lookup and rule-based approaches. Jabbar et al. [16] proposed a multi-step Urdu stemmer that works in two phases: In the first phase, compound words are extracted from the text fragment using punctuation marks as well as stop words as a delimiter, for instance, the Urdu sentence وہ تعلیم یافتہ ہے۔ [he/she is educated], after removing the punctuation marks and stop words, compound word تعلیم یافتہ [educated] is retrieved and the suffix is removed یافتہ,, the extracted stem is تعلیم [education]. In phase two, unigram words that require recoding (substitution, deletion, and/or addition) are used to strip off the suffixes such as the suffix گی is replaced by ہ, for example, the Urdu word بندگی [worship] stems to بندہ [worshipper]. The circumfixes (both prefix and suffix), prefix and suffix are treated such as ناخوشگوار [unpleasantness] stems to خوش [pleasant] by removing the prefix نا and suffix گوار.. If the query word contains only a prefix or suffix, then apply the appropriate rules to derive the stem as the Urdu word بقلم [with pen] in which ب is a prefix and قلم [pen] is a stem. After this infix is traced using a predefined pattern and is truncated to get the stem, for instance, احکام [orders] stem to حکم [order]. In the last, if the stem is not derived in the prior steps, the table lookup the relevant stem, otherwise original word is returned. Mahalingam, [92] presented a stemmer named Bruteporter: A hybrid stemmer for the English language, that used Wordnet, and Modified Porter [38] Algorithm. First, the modified Porter [38] rules are applied to the query word, if no match is found then Wordnet is used to derive the stem. Amin et al. [93] proposed a stemmer that consists of rule-base and string similarity approaches. In these stemming methods, the first rules are applied to retrieve the stem, if no stem is retrieved then the string is matched with database entries, and a database string with the highest similarity is taken as the possible stem.

Taghi-Zadeh et al. [94] proposed a statistical stemmer that mainly consists of two stages. Stage one is concerned with building the affix list automatically from the training corpus and the second stage is related to extracting the stem from the query word. In the first stage candidate affixes are generated using tri data structure, in which the root is empty. Fig. 5 depicts the tri data structure for three words 'awed', 'axed', and' aced' and an inverted tree is shown in Fig. 6, in which common suffixes from the query word are taken. Every path in the inverted tree suggested an affix and stem, 'Cut-point' indicates the node from where the affix and stem are separated from each other. Several strategies are described in which the maximum score strategy is adopted. The description of this strategy is given below.

$$i_{cutpoint} = \arg\max Score\,(i_x) \quad x \in [1...,k]$$

$(i_1, i_2, ...,i_K)$ is a cut point in the path.

Suffixes generated in this step are doubted so to confirm, these affixes are passed to the affix candidate filtering process in which K-mean clustering is used to distinguish the valid and invalid affix, $k = 2$). After this, the valid affix is transformed into a regular expression form.

In the second stage, the query word is searched in a dictionary according to the POS, and the relevance of POS stemming rules is applied. Another choice is based on predefined pattern matching to derive the stem.

Selvaramalakshmi et al. [49] presented another stemming method that functions in three stages either linear or nonlinear. The first step prunes the affixes (prefixes and suffixes) using the predefined prefix and suffix list after this extracted stem is compared with WordNet and documented multiple matching in the form of Multisets (Mi). SSize filtration Eq. 19 determined the similarity of two strings if the value of SSize is equal to a predefined threshold. If the pair $\{M_i, M_j\}$ satisfies the condition $|M_j| \geq \gamma^* |M_i|$ then it is passed to the filter, if not, then pruned.

$$SSize = \frac{\sum_{i=1}^{n} |M_i| \times \gamma}{Total\ No.of\ sets} \tag{19}$$

The second step is concerned with the positional filtration that is calculated by the Jaccard Equation (Eq. 19). The third step is to reverse the pair $\{M_j, M_i\}$ to retrieve the value of M_j,
Mateen et al. [108]

developed a hybrid stemmer for the Punjabi language, incorporating rule-based and table-lookup approaches. First table lookup technique is used to obtain the stem, if the stem is not found, then a rule-based approach is applied to derive the stem. Momenipour and Keyvanpour [95] proposed a stemmer for the Persian language that first checks the given word in the stem-word dictionary, if found then its corresponding word is returned as the stem, otherwise, HMM is used to determine the possible affix (prefix and suffix) and are removed. Sitaula, [96] defined the Nepali

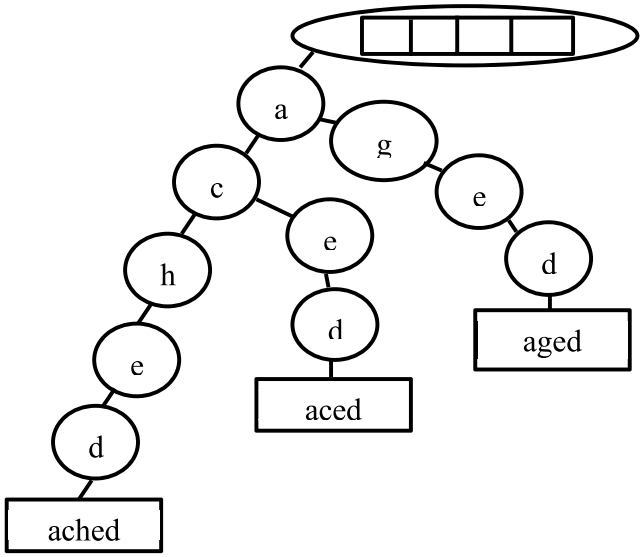


FIGURE 5. Example of a trie data structure.

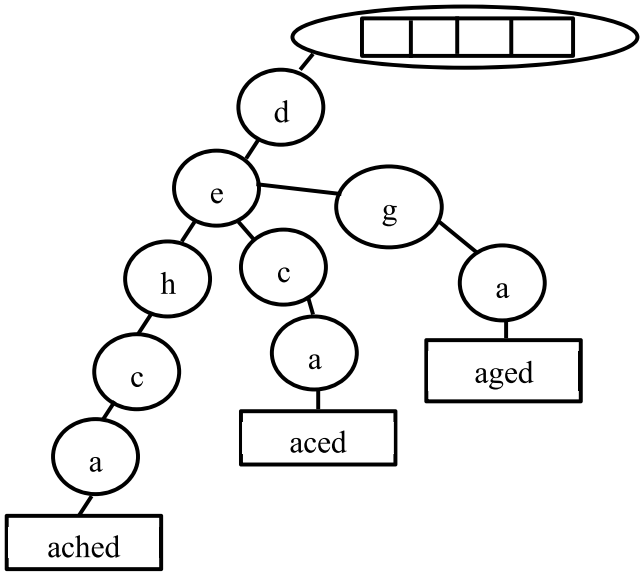


FIGURE 6. An inverted trie examples.

stemmer using the traditional rule-based approach with an edit distance measure to keep the threshold value 0.5. Saharia et al. [90] incorporate the rule-based approach with HMM to develop a stemmer for Indian-resourced poor languages that include Assamese, Bengali, Bishnupriya Manipuri, and Bodo. Majgaonker and Siddiqui [68] used the rule-based technique and N-gram techniques to develop a stemmer for the Marathi language. Through N-gram automatically extracted the suffix from the training corpus, and manually suffix list built-in rule-based approach.

F. COMPARISON OF STEMMING TECHNIQUES

This section describes, compares, and summarizes the recent advances in stemming techniques. Table 13 highlights the strengths and weaknesses of the current stemming methods.

TABLE 13. Strengths and weakness of text-stemming approaches.

TS Method	Strengths	Weaknesses
Manual stemming	Manual TS decreases the error rate.	Time time-consuming and inefficient for a large corpus.
Linguistic knowledge-based approaches	Effective in terms of IR and vocabulary compression.	Did not consider the context of the word and, as a result, produced an irrelevant stem.
Affix striping	This stemmer will give an excellent result based on excellent rules.	• Affix (prefix and or suffix) list which is the termination part of the word in many languages, but it is not suitable for morphologically complex languages like Arabic. • Performance decreases in case of irregular query words. • Too many rules make it more complex, and some rules are opposite to each other in that situation rule base stemmer will not give better results. A rule is required to be removed for applying an opposite rule. • Longest affix matching may suffer over stemming error, in contrast, iterative affix striping is time-consuming.
Brute force/ Table lookup	The retrieve stem is always a correct word which is found in the table, so the lowest error rate. Resolve the polysemy and synonymy issues.	• It is language as well as data-dependent, which is in the table. • Constructing and updating required a ridiculously human laborer. • A large space is required to store the table. • Low recall and precision
Template/ Pattern matching	Avoid the complexity of the rules.	• A language expert is essential to build templates and extract patterns from the underlying language. • Inflection makes the conflation process more challenging, such as write, wrote, and written are complex verbs of the English language, which do not follow the normal process to change the verb form. • Some words are patternless.
statistical-based approaches	• It can be used in a Multi-lingual NLP system. • Give better results in the IR system.	• Computationally expensive • Suitable only suffixing languages • Sometimes wrongly consider semantic relations. • Corpus dependent • Threshold value setting issue • Statistical stemmer unable to handle exceptional cases such as mouse>>mice, feet >> foot
N-gram based approaches	No linguistic knowledge is required. Based on N-gram and string	• The algorithm required a large space to store the N-gram tokens and the indexes. • Time-taking algorithm. • Stem length is limited to four characters.
	similarity approaches.	• It may commit over stemming and under stemming errors.
HMM-based approaches	Language-independent and Unsupervised HMM-based models better perform in many NLP applications, regardless of the context of the language.	• It may produce over-stemming errors. • Hard to understand and implement, for at every node some probability function is applied. • Time inefficient. • Only handle suffixes.

G. EVALUATION METRICS

In the literature, several stemming performance evaluation methods have been carried out to check the strength of a stemmer for finding the correct stem. Jabbar et al. [89] imparted the analyses of present TS evaluation methods.

TABLE 13. *(Continued.)* **Strengths and weakness of text-stemming approaches.**

Corpus-based approaches	• Applicable to a wide variety of languages. • Improved the performance of IR systems.	• Exceptional cases such as mouse>>mice, feet >> foot are not handled. • Setting the threshold value is difficult. • Requires a large and wide variety of corpora to train the system. • Computationally expensive because the same word may have many synonyms and similar forms. • Requires more processing time as well as a significant amount of memory.
Context-sensitive based approaches	• The context-sensitive strategy avoids spurious stemming. • Better query expansion that is conservative to relevant word matching, consequently, reducing computation and improving the accuracy.	• Complex and time-consuming algorithm. • The correctness of stem relies on the context information. • Language dependent.
Hybrid based approaches	Incorporating more than one stemming approach may overcome the limitations of each other.	• Some combination may increase the computation and • If dictionary lookup is combined, then extra overhead for storage.

Some of the popular evaluation mechanisms have been investigated in this section.

1) CONFLATION BASED EVALUATION

Paice [90] proposed a few attributes to measure the performance of a stemmer which include the over-stemming index (OI), under-stemming index (UI), the stemming weight (SW), and an error rate relative to truncation (ERRT). According to Paice [90], the stemming weight $SW = OI/(UI)$ indicates the strength of the stemmer. Lovins [59] Porter [38] and Paice/Husk [97] stemmers are evaluated and it is found that Paice/Husk stemmers are the strongest and Porter is the weakest among experimented stemmers. Frakes and Fox [98] presented Modified Hamming Distance statistics to evaluate stemmers. They experimented with Lovens, Paice, Porter, and S-removal methods and claimed that Paice/Husk stemmer is stronger than Lovins, and Lovins is stronger than porter. Sirsat et al. [99] also proposed some metrics to measure the performance of a stemmer: Index compression factor (ICF), Word Stemmed Factor (WSF), Correctly Stemmed words Factor (CSWF), and Average Words Conflation Factor (AWCF). They experimented with the Lovins Stemmer, Porter1 Stemmer, Porter2 Stemmer, and Paice Husk Stemmer on a data set of 1858 words. The porter2 algorithm is an improved version of the Porter base version [50] stemmer, which is available at.[1] The highest ICF is 64.63 obtained by the Lovins [55] and Porter1 achieved the lowest ICF which is 51.88. The AWCF score of 19.26, the highest among others. AWCF can be calculated using the following formula.

$$NWC = S - CW$$

where NWC is the number of distinct words counted after conflation, S is the number of distinct stems and CW is the number of correct words which are not stemmed. If CSW is the number of correctly stemmed words, then AWCF is calculated as follows:

$$AWCF = \frac{CSW - NWC}{CSW} * 100$$

The AWCF score obtained by Lovins and Porter1 stemmer is negative which shows that a greater number of words were wrongly stemmed than correctly stemmed words [99]. Lovins [55] stemmer is more aggressive than others as WSF obtained is highest 73.35, but CSWF is 27.80 is lowest. The highest CSWF was 34.76 achieved by Porter2.

2) INFORMATION RETRIEVAL (IR) BASED EVALUATION

When IR systems utilize the stemming as a preprocessing step, the performance of the stemmer is measured by Precision, Recall, F-measure, Average precision (AvP), and mean average precision (MAP), however, these are extremely reliant on other processes of IR systems. Can et al. [65] proved that stemming can improve the performance of IR systems and in this context, the authors have used the Turkish IR system as an example to achieve 38% performance improvement.

3) TEXT CLASSIFICATION-BASED EVALUATION

Stemming is used as a preprocessing step in feature extraction for text classification systems [94], [95] which are briefly explained below:

K-Nearest Neighbor (K-NN) classifier: This algorithm compares the test and training datasets by considering both the value of **K** and the similarity measure. Subsequently, it arranges the data in descending order. This type of classifier is commonly referred to as a lazy classifier [100].

Naïve Baye Classifier: Naïve Bayesian Classifier is a probabilistic classifier that assumes that the features used for classification are conditionally independent. This algorithm computes different statistical parameters like Mean (μ) and Variance (σ) for each class given in the dataset. Usually, low variance shows the better performance of the system [97].

Support Vector Machine (SVM) Classifier: SVM is a linear, non-probabilistic, and supervised learning classifier that constructs a linear hyper-plane that splits the data into two i.e. positive and negative classes. The SVM deals with high-dimensional data, without falling into over-fitting problems. The main function of SVM is to divide the instances into two groups with maximum margin.

[1] http://snowball.tartarus.org/algorithms/english/stemmer.html

Maximum Entropy (MaxEnt) Classifier: It is a probabilistic exponential classification that is frequently utilized in language processing programs such as sentiment analysis. Almuzaini & Azmi [100] used a Maximum Entropy classifier to assess the performance of the proposed stemmer for the Kannada language.

TABLE 14. Performance comparison of rule-based stemmers.

Ref.	Tested language	Dataset Size (Words)	Performance metric/s	Results %
[19]	GJ	485949	P, R, F	0.97
[101]	SN	1000	Manual Method	60
[40]	NP	1400 news articles	F	0.79
[42]	AR	PATB	P, R, F	97.34
[102]	PB	3,135	Manual Method	95.21
[19]	GJ	2197	Manual Method	98.33
[45]	KR	KDC-4007[2]	P, R, F	87
[103]	SD	4,453	Manual Method	96.79
[44]	UR	32000	P, R, F	85.02
[104]	ML	2028 documents	Manual Method	90.37
[105]	IND	379	Manual Method	88.65
[83]	EN	The Web Track Two-Gigabyte (WT2G) dataset	P, R, F	99
[48]	AR	ARASTEM dataset[3]	Paice, (1994, 1996) 's Evaluation Method	UI= 0.618 OI=0 SW=0
[106]	MR	99,275 documents	Manual Method Frakes and Fox (2003) evaluation method	79.97
[49]	EN	Not given	Manual Method	94.63
[50]	AR	1000 words	Manual Method	79.6
[41]	ML	58,563	Manual Method	98.34
[107]	GJ	6530	Manual Method	96.63
[108]	TK	2,468 law entries	P	25% improvement

**Arabic (AR), Gujrati (GJ), Sindhi (SN), Nepali (NP), Punjabi (PB), Sudanese (SD), Urdu (UR), Malay (ML), Indonesian (IDN), English (EN), Marathi (MR), Turkish (TK), Kurdish (KR), Persian (PR), Czech (CZ), Slovak (SL), Polish (PL), Hungarian (HU), Spanish (SP), Italian (IT), French (FR), Latin (LT), Portuguese (PR), Hindi (HN)*
*** Precision (P), Recall (R), F-measure (F)*

V. PERFORMANCE COMPARISON AND CRITICAL ANALYSIS

This section focuses on comparative analysis of claimed results, used data sets and evaluation mechanisms among different TS systems. A critical analysis was also performed to highlight the major issues in the current TS algorithms. Generally, linguistic knowledge-based stemmers perform well because they follow linguistic rules and are evaluated mostly using the gold standard evaluation method as depicted from Table 14. On the other hand, statistical stemmer tests on large data sets and, usually, evaluated in the IR system as shown in Table 15. Corpus-based stemmer is also evaluated in IR systems and using large dataset size as mentioned in Table 16. Context-sensitive stemmers are tested on the large size of data and evaluated by precision and recall as mentioned in Table 17. Table 18 compares the claimed accuracy, evaluation mechanisms, and data sets of the hybrid stemmers. Mostly, the hybrid stemmers were evaluated by the manual method.

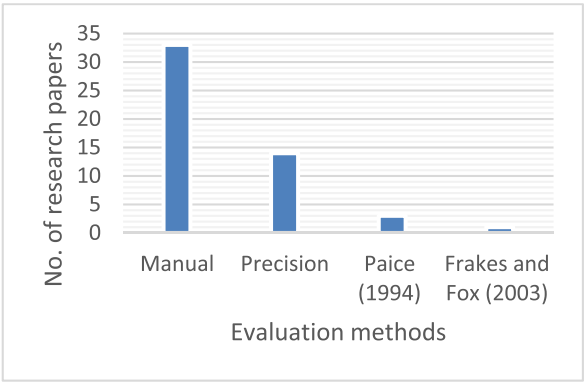


FIGURE 7. Overview of used evaluation methods by linguistic knowledge-based stemmers.

Mostly, the linguistic knowledge-based stemmers are evaluated by a manual means as depicted in Fig. 7. To analysis the used evaluation datasets, the authors defined the following criteria for large, medium, and small data sets.

- Large: if the number of Words greater than or equal to 20000
- Medium: when the number of words is between 5000 and 20000.
- Small: Number of words less than 5000

Fig. 8 shows that 52% of linguistic knowledge-based stemmers are evaluated on large datasets, 35% on medium, and 13% on small datasets. Mezzi et al. [83] claimed the highest accuracy 99% on English text as mentioned in Table 12. The performance of a stemmer may vary from language to language because every language has its grammatical structure. Such as For the Arabic language highest performance is 97.34 achieved by [42]. Obtained results may also vary if the evaluation dataset size is increased. Hence, the state-of-the-art current stemming algorithms are developed through small training datasets, so those approaches are not applicable in a wide prospective. As a result, a huge dataset is required to obtain high accuracy.

Table 15 shows that the language-independent stemmers are generally evaluated in more than one language such as [64] tested the stemmer in Czech, Slovak, Polish, Hungarian, Spanish, and English language. Ferilli et al. [65] tested the stemmer on English, Italian, French, and Latin text. Statistical stemmers, evaluated in the context of IR systems such as Dadashkarimi et al. [63] stemmer are shown in Table 15. Statistical stemmer's performance may vary from language to language as mentioned in Table 15, for example,

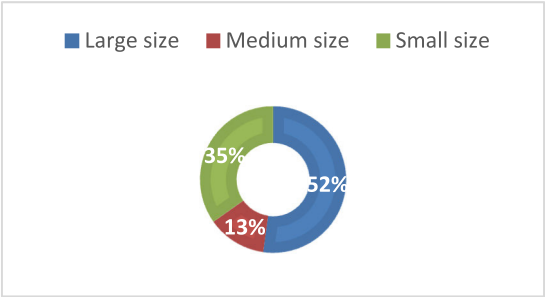


FIGURE 8. Overview of evaluation dataset used by linguistic knowledge-based stemmers.

TABLE 15. Performance comparison of statistical stemmer.

Ref	Tested language	Data set	Perform. metric	Results %
[61]	BG	46892 words	Manual	87
[63]	PR	Hamshahri 1996-2002	MAP	0.268
[64]	1. CZ 2. SL 3. PL 4. HU 5. SP 6. EN	1. (news corpus[4]) 2. Slovak National Corpus5) 3. Multilingual parallel corpus (MPC) 4. MPC[6] 5. Reuters Multilingual Corpus (RCV2)[7] 6. Los Angeles Times[8]	P R F	F-measure[9] CZ: 53.4 SL: 57.5 PL: 46.1 HU: 52.5 SP: 49.7 EN: 70.8
[65]	EN IT FR LT	EN: 11053 words IT: 24008 words FR: 20099 words LT: 13788 words	ICF	EN: 91.17 IT: 92.63 FR: 89.54 LT: 92.78
[67]	MR UR	MR:1200 words UR:1200 words	Manual	MR: 63.52[10], 82.68[11] UR: 85.36[12], 81.28[13]
[69]	EN PT AR	EN: 3,975 words PR: 120 words	P, R, F	F: 85
[68]	HI	EMILLE corpus	P, R, F	F: 94.96
[70]	AR	2000 words	Manual	Cut off method. 80 Entropy method 75

a stemmer has achieved 92.63% accuracy for the Italian language whereas it achieved 89.54 for the French language. From Table 15, most statistical stemmer evaluated on a large dataset.

Corpus-based stemmers are also tested for multiple languages (see Table 16) such as Singh and Vishal Gupta [109] tested their stemmer for English, Marathi, Hungarian, and Bengali languages. This type of stemmer is evaluated in the IR systems. The performance may vary from language to language for instance, [76] obtained the highest MAP in Marathi 0.451 and the lowest in English 0.270.

Table 16 shows all the corpus-based stemmer tested on large dataset.

TABLE 16. Performance comparison of corpus-based stemmers.

Ref	Tested lang.	Data set	Perform. metric	Results %
[109]	EN MR HU BG	TREC, CLEF, FIRE standard test collections.	MAP	EN: 0.3084 MR: 0.3804 HU: 0.3341 BG: 0.3383
[73]	EN MR HU BG	WSJ documents (EN) Sakal and Maharashtra Times (MR) Magyar Hirlap corpus (HU) FIRE[14] 2010 collection (BG)	MAP	EN: 0.323 MR: 0.410 HU: 0.324 BG: 0.330
[74]	EN MR HU BG	173,252 WSJ documents (EN) 99,275 documents (MR) Magyar Hirlap collection of 49,530 documents (HU) FIRE 2010 collection containing 123,047 documents (BG)	MAP	EN: 0.3236 MR: 0.4184 HU: 0.3229 BG: 0.3314
[110]	EN MR HU BG CZ BL FR	No. of documents MR: 99362 HU: 49530 CZ: 81735 EN: 472525 BL: 87281 BG: 123047 FR: 177452	MAP	MR: 0.451 HU: 0.351 CZ: 0.366 EN: 0.270 BL: 0.326 BG: 0.476 FR: 0.387
[111]	EN MR HU BG	EN: 528315 MR: 533605 HU: 854324 BG:179387	MAP	MR: 0.386 HU: 0.341 EN: 0.295 BG: 0.387

* EN (EN), Marathi (MR), Hungarian (HU), Bengali (BG), Czech (CS), Bulgarian (BL), French (FR)

It is shown in Table 17, some context-sensitive stemmers are tested for more than one language such as [31] proposed stemmer which is tested on Turkish, Hungarian, Finnish, Basque, and English. On the other hand, some evaluated only one language for instance [82] tested on English, Al-Shammari et al. [87] tested on Arabic, and Sarkar and Bandyopadhyay [90] on Bengali language. The highest performance is 96.7% obtained by Adam et al. [86] on Greek text. Mostly context sensitive stemmers are tested on large datasets as shown in Table 17.

Generally, hybrid stemmers are language-specific and are tested on a specific language as shown in Table 18. [91] tested the stemmer in the Marathi language. Jabbar et al. [16] designed a stemmer that obtained a 96.26% F-score.

The different stemming algorithms are quite similar in their objectives, but none of them give 100% output. As mentioned in Fig. 9 linguistic knowledge-based approach is widely used. Rule-based algorithms provide the highest accuracy 99 % among all the existing algorithms as shown in Table 14.

TABLE 17. Performance comparison of context sensitive stemmers.

Ref	Methods	Tested lang.	Data set	Perfor. metric	Results %
[31]	HMM Model	TR HU FI BA EN	TK: 5620 sentences and 53798 tokens HU: 24K words FI: 19000 sentences and 162000 words BA: 24K words EN: 24K words	Frakes and Fox (2003)	TK: 63.83 HU: 70.12 FI: 45.40 BA: 55.50 EN: 83.84
[82]	Word Embed.	EN	100K EN tweets posted	MAP R	0.59
[31]	HMM Model	TK EN FI	TK: METU-Sabancı Turkish Treebank EN: WSJ Penn Treebank FI: FinnTreeBank[15] corpus	Variation of Information (VI) measure	TK: 46.57[16] FI: 27.95[17]
[85]	Sussif removal algorithms	EN	10000 words	Manual Method, P R F	Error rate: 0.29

There is neither a standard dataset nor standard evaluation methods. The evaluation comparison of the TS algorithms shows that TS is still one of the most challenging tasks.

VI. PERFORMANCE OF A TEXT STEMMER

Several factors affect the performance of the TS systems such as types of language, domain, and types of affixes to be handled. Some languages are resource-poor, consequently making the TS task challenging. As the types of affixes increase the complexity of the system is also increased because more rules are coined to identify the affix types.

A. LANGUAGE-SPECIFIC FACTOR

A language can be either concatenative or non-conatenative languages. Specific rules are utilized to determine the stems in concatenative languages, while in non-conatenative languages, affixes and stems are intertwined. Non-concatenative morphology, which typifies Semitic languages like Arabic, and Hebrew, in which words are formed in the form of vocalic and consonantal patterns [113]. Developing a stemming system task becomes harder as the morphology and grammar. Orthography and character encoding of the goal language become more multifaceted, generally, it is the inflection of the language. For example, Italian languages are morphologically richer, having more possible verb inflections than English, the Russian language is also more complex and includes a neutral gender, and Arabic is even more complex, for broken plural, dual count plural, and more than two plural and it has also infixed. In the German language, a verb may have 144 forms with multiple derivational inflections [114].

TABLE 18. Performance comparison of hybrid stemmers.

Ref	Methods	Tested lang.	Data set	Perfor. metric	Results %
[91]	Rule-based and table-lookup	MR	4500 terms	Manual Method	ACC89.8
[16]	Affix striping, template matching, and table lookup	UR	76074 words	P R F ICF	F: 96.26
[92]	Rule-based and table-lookup	EN	1000 words	Manual Method	93.4
[94]	Rule-based and statistical	PR	975,000 words	Manual Method	88.40
[112]	Rule-based and table-lookup	PU	2.5 million tokens	Manual Method	86.01
[93]	Rule-based and string similarity	Ngoko Javanese language	2631 words	P R F	67
[95]	Table Lookup and HMM	PR	500	Manual Method	79

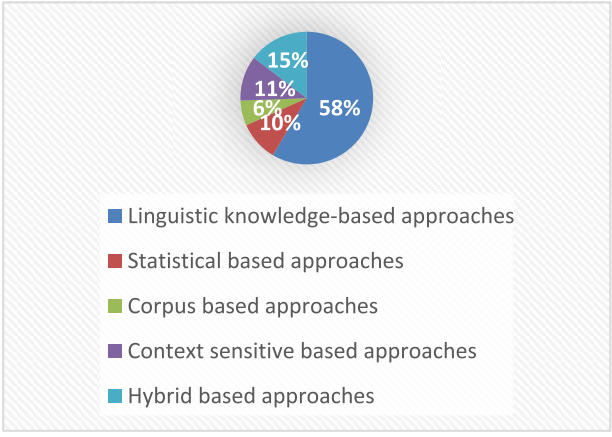


FIGURE 9. Comparison of types of TS approaches.

In the Urdu language, 57 variant forms can be generated from a single Urdu word [115].

The English language is considered a weakly inflective language [116]. So, it is a very easy step to obtain a stem because of only suffix removal [38]. In contrast, in the Indonesian language a wide range of affixes as prefixes,

suffixes, confixes, and suffixes. The Arabic language also has multiple affixes [116]. Types of affixes may vary from language to language, usually, a language has a prefix and suffix affixes, but some languages have many types of affixes, such as Sundanese is a rich morphological language that has at least 54 main affix types [103]. To ensure efficient retrieval all these forms should be conflated to the same stem.

B. DOMAIN AND DATA DEPENDENCY FACTOR

Normally, a stemmer is designed to process general texts, but table lookup/brute force approaches and corpus-based approaches render domain-specific content. Though the table lookup/brute force approach is accurate however it is restricted to the words listed in the table making stemmer domain specific. In the corpus-based approach, word frequency and/or various co-occurrence statistics are used instead of the grammatical and morphological set of rules that make it domain-specific.

The size and type of training and test dataset directly influence the performance of the corpus-based and statistical stemmer. Linguistic knowledge-based stemmer is less affected by types of documents used in rules development than the other types of stemmers. Moral et al. [10] related the performance of the stemmer to the size of the test data set. Moghadam and Keyvanpour [9] claimed that the obtained results may vary with the increase in the size of the dataset.

C. TASK FACTOR

Some stemmers are developed for specific NLP applications such as [73] constructed stemmer for IR systems, Boukhalfa et al. [75] stemmer developed for the improvement of the accuracy of plagiarism detection, and Ali et al. [47] developed stemmer for text classifications. The performance of a stemmer may vary from one NLP application to another. Linguistic knowledge-based stemmer performs better in IR systems and text classification. Context-sensitive stemmers outperform IR systems as compared to other types of stemmers.

VII. CURRENT ISSUES AND CHALLENGES

A plethora of research has already been carried out in stemming. However, still, there is a requirement to develop better algorithms for general and specific stemming. There are specific aspects that make stemmer design more challenging such as recognizing proper nouns, stemming compound words, exception handling, homographs recognizing, annotation of the dataset, processing complexity, and computational complexity. These issues should be considered carefully during the development of the TS system. The following text provides more details about such issues.

Conflation is a reverse process of grammatical rules and almost every grammar rule has some exception, for instance, in English ''ed'' is used as a suffix in most cases, but in some cases, it is an actual part of the word such as the English word ''red''. In the Portuguese language suffix, ''âo'' denotes augmentative, however not all words having the suffix ''âo'' are augmentative [117].

All the natural languages contain homographs which means the words with the same spelling may change meaning according to the context, consequently, the root is also different. Homographs are presented in European languages as well as in Indo-European language families [118] and create obstacles to retrieving the relevant root word. French languages are Slavic languages, which are rich in homographs. Proper nouns reduce the performance of many NLP applications [34]. Proper nouns should not be stemmed [119], however, if these are stemmed by the algorithm, reduce the performance of the stemmer. Reference [120] demonstrated that decompounding German words could significantly improve the performance of IR systems. A compound word may contain a hard space, a hyphen, or be solid. A hyphen is a strong clue for identifying compound words in most European languages such as in French [porte-clefs/ key ring].

Almost all TS approaches increase the processing cost. Many rules in knowledge-base and preprocessing of the large corpus in statistical (n-gram) and corpus-based systems are examples of increased processing. In the case of a knowledge base system, a large amount of knowledge is required to develop rules. So, the developer needs to optimize the processing complexity. There is a tradeoff between the accuracy and execution time of a stemmer [121]. Usually, expensive mathematical calculations are performed to extract various features, in corpus-based and statistical approaches, which increase the overall complexity of the text-stemming system.

Generally, gazetteers and annotated corpus are used to develop and evaluate the performance of a stemmer. Similarly, Named Entity Recognition (NER) systems, POS Tagger, Morphological Analyzer, and word segmentation systems also play a prominent role in designing a TS system. Some languages are classified as resource-poor, for instance, Mongolian, Indonesian, Hindi, Urdu, Sindhi, Punjabi, Pashto, and Bengali, hence, making the TS for such languages is more challenging. The labeled/ annotated data are essential to design and test stemmers. However, labeling the data set is a difficult and time-consuming task and requires domain experts to correctly annotate it.

Considering the available stemming methods, it is a complex question to select the right method for the application under consideration. A few parameters need to be analyzed before selecting the right stemmer which may include the nature of language, size of data, and the nature of the problems being investigated. As an example, for a resource-poor and complex language, the linguistic knowledge-based approach is a good choice, or hybrid approaches can produce good results. Existing statistical and corpus-based stemmers consider only suffixes and ignore the prefixes and infixes. So, it is required to extract more features that can handle the prefixes and infixes too. Further, a language-independent and auto-feature extraction mechanism is required to develop a high-performing TS system.

Besides all these, the TS evaluation method and test data collection are major factors that can change the obtained accuracy.

Usually, TS systems are applied or designed according to NLP applications. There is a requirement to design an evaluation mechanism that can work universally. Some direct evaluation methods use the compression index factor, under stemming errors stemming errors, etc. Mostly linguistic knowledge-based stemmers are evaluated on a small dataset, hence there is the need to test design methods on large datasets.

VIII. CONCLUSION

This article presents a comprehensive review of linguistic knowledge-based, statistical, corpus-based, and hybrid stemmers. Moreover, it carries out an extensive comparative analysis of the above-mentioned approaches. It is observed that the linguistic knowledge-based stemming technique is widely used for highly inflected languages (such as Arabic, Hindi, and Urdu) and has reported more accuracy than the other techniques. Likewise, exception handling and homograph recognition are challenging tasks for linguistic knowledge-based stemmers. Similarly, semantic meaning-extraction, affix stripping rules self-construction, and auto parameter tuning are major issues for unsupervised stemming techniques. The selection of the TS evaluation method is still an open question. While analyzing the existing techniques, it is found that the manual evaluation methods for a stemmer performance exhibit dissimilar behavior while testing on small and large datasets. As described earlier, the researchers must consider certain factors affecting the performance of text stemmers (such as language dependency, domain, and data dependency, and task factors) while designing and developing a stemmer.

Consequently, we recommend that the researchers use different machine learning algorithms to acquire the semantic meaning of the query word and striping part. Due to the varied and heterogeneous nature and size of the datasets and knowledge bases, selecting the appropriate stemming technique is tedious. It requires deep investigation by the research community. In the future, we aim to utilize TS as an intelligent agent to extract semantic stem from a user query without requiring an explicit request.

FUNDING

No specific funding received for this research.

ACKNOWLEDGMENT

The authors would like to thank the AIDA Laboratory, CCIS, Prince Sultan University, Riyadh Saudi Arabia, for their support.

DECLARATION

Authors declare no conflict for this research.

REFERENCES

[1] S. U. Hassan, J. Ahamed, and K. Ahmad, "Analytics of machine learning-based algorithms for text classification," *Sustain. Oper. Comput.*, vol. 3, pp. 238–248, Jan. 2022, doi: 10.1016/j.susoc.2022.03.001.

[2] A. Jabbar, S. U. Islam, S. Hussain, A. Akhunzada, and M. Ilahi, "A comparative review of Urdu stemmers: Approaches and challenges," *Comput. Sci. Rev.*, vol. 34, Nov. 2019, Art. no. 100195, doi: 10.1016/j.cosrev.2019.100195.

[3] A. Jabbar, S. Iqbal, M. U. G. Khan, and S. Hussain, "A survey on Urdu and Urdu like language stemmers and stemming techniques," *Artif. Intell. Rev.*, vol. 49, no. 3, pp. 339–373, Mar. 2018, doi: 10.1007/s10462-016-9527-1.

[4] M. Mustafa, A. S. Eldeen, S. Bani-Ahmad, and A. O. Elfaki, "A comparative survey on Arabic stemming: Approaches and challenges," *Intell. Inf. Manage.*, vol. 9, no. 2, pp. 39–67, 2017, doi: 10.4236/iim.2017.92003.

[5] J. Singh and V. Gupta, "A systematic review of text stemming techniques," *Artif. Intell. Rev.*, vol. 48, pp. 157–217, Aug. 2017, doi: 10.1007/s10462-016-9498-2.

[6] J. Singh and V. Gupta, "Text stemming: Approaches, applications, and challenges," *ACM Comput. Surv.*, vol. 49, no. 3, pp. 1–46, Sep. 2017.

[7] M. Y. Dahab, A. Ibrahim, and R. Al-Mutawa, "A comparative study on Arabic stemmers," *Int. J. Comput. Appl.*, vol. 125, no. 8, pp. 38–47, Sep. 2015.

[8] Z. F. Muhsin, A. Rehman, A. Altameem, T. Saba, and M. Uddin, "Improved quadtree image segmentation approach to region information," *Imag. Sci. J.*, vol. 62, no. 1, pp. 56–62, Jan. 2014.

[9] F. M. Moghadam and M. R. Keyvanpour, "Comparative study of various Persian stemmers in the field of information retrieval," *J. Inf. Process. Syst.*, vol. 11, no. 3, pp. 450–464, 2015, doi: 10.3745/JIPS.04.0020.

[10] C. Moral, A. de Antonio, R. Imbert, and J. Ramírez, "A survey of stemming algorithms in information retrieval," *Inf. Res.*, vol. 19, no. 1, p. n1, 2014.

[11] M. A. Otair, "Comparative analysis of Arabic stemming algorithms," *Int. J. Manag. Inf. Technol.*, vol. 5, no. 2, pp. 1–12, May 2013, doi: 10.5121/ijmit.2013.5201.

[12] V. Gupta and G. S. Lehal, "A survey of common stemming techniques and existing stemmers for Indian languages," *J. Emerg. Technol. Web Intell.*, vol. 5, no. 2, pp. 157–161, May 2013, doi: 10.4304/jetwi.5.2.157-161.

[13] U. Sharif, Z. Mehmood, T. Mahmood, M. A. Javid, A. Rehman, and T. Saba, "Scene analysis and search using local features and support vector machine for effective content-based image retrieval," *Artif. Intell. Rev.*, vol. 52, no. 2, pp. 901–925, Aug. 2019.

[14] A. Jabbar, S. Iqbal, M. I. Tamimy, S. Hussain, and A. Akhunzada, "Empirical evaluation and study of text stemming algorithms," *Artif. Intell. Rev.*, vol. 53, no. 8, pp. 5559–5588, Dec. 2020, doi: 10.1007/s10462-020-09828-3.

[15] T. Kanan, O. Sadaqa, A. Almhirat, and E. Kanan, "Arabic light stemming: A comparative study between P-stemmer, Khoja Stemmer, and light10 stemmer," in *Proc. 6th Int. Conf. Social Netw. Anal., Manage. Secur. (SNAMS)*, Oct. 2019, pp. 511–515, doi: 10.1109/SNAMS.2019.8931842.

[16] A. Jabbar, S. Iqbal, A. Akhunzada, and Q. Abbas, "An improved Urdu stemming algorithm for text mining based on multi-step hybrid approach," *J. Exp. Theor. Artif. Intell.*, vol. 30, no. 5, pp. 1–21, May 2018, doi: 10.1080/0952813x.2018.1467495.

[17] N. Madi, N. Al-Mutlaq, and H. S. Al-Khalifa, "HealthSEA: Towards improving the search engine of KAAHE Arabic health encyclopedia," in *Proc. 2nd Int. Conf. Comput. Appl. Inf. Secur. (ICCAIS)*, May 2019, pp. 1–7.

[18] N. Yusuf, M. A. M. Yunus, and N. Wahid, "Arabic text stemming using query expansion method," in *Emerging Trends in Intelligent Computing and Informatics* (Advances in Intelligent Systems and Computing). Johor, Malaysia: Springer, 2020, pp. 3–11, doi: 10.1007/978-3-030-33582-3_1.

[19] C. D. Patel and J. M. Patel, "Influence of Gujarati STEmmeR in supervised learning of web page categorization," *Int. J. Intell. Syst. Appl.*, vol. 13, no. 3, pp. 23–34, Jun. 2021, doi: 10.5815/ijisa.2021.03.03.

[20] M. S. Mau?ec and G. Donaj, "Machine translation and the evaluation of its quality," in *Recent Trends in Computational Intelligence*, vol. 143. London, U.K.: IntechOpen, 2019.

[21] R. Elbarougy, G. Behery, and A. El Khatib, "Extractive Arabic text summarization using modified PageRank algorithm," *Egyptian Informat. J.*, vol. 21, no. 2, pp. 73–81, Jul. 2020.

[22] A. R. Pathak, M. Pandey, and S. Rautaray, "Topic-level sentiment analysis of social media data using deep learning," *Appl. Soft Comput.*, vol. 108, Sep. 2021, Art. no. 107440.

[23] A. F. Anees, A. Shaikh, A. Shaikh, and S. Shaikh, "Survey paper on sentiment analysis: Techniques and challenges," in *Proc. EasyChair*, 2020, pp. 1–5.

[24] S. Al-Saqqa, A. Awajan, and S. Ghoul, "Stemming effects on sentiment analysis using large Arabic multi-domain resources," in *Proc. 6th Int. Conf. Social Netw. Anal., Manage. Secur. (SNAMS)*, Oct. 2019, pp. 211–216.
[25] I. Budi and R. R. Suryono, "Application of named entity recognition method for Indonesian datasets: A review," *Bull. Electr. Eng. Informat.*, vol. 12, no. 2, pp. 969–978, Apr. 2023.
[26] M. Garriga, A. D. Renzis, I. Lizarralde, A. Flores, C. Mateos, A. Cechich, and A. Zunino, "A structural-semantic web service selection approach to improve retrievability of web services," *Inf. Syst. Frontiers*, vol. 20, no. 6, pp. 1319–1344, Dec. 2018.
[27] S. K. Ray and K. Shaalan, "A review and future perspectives of Arabic question answering systems," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 12, pp. 3169–3190, Dec. 2016.
[28] Y. A. Alhaj, J. Xiang, D. Zhao, M. A. A. Al-Qaness, M. A. Elaziz, and A. Dahou, "A study of the effects of stemming strategies on Arabic document classification," *IEEE Access*, vol. 7, pp. 32664–32671, 2019, doi: 10.1109/ACCESS.2019.2903331.
[29] J. O. Atoum and M. Nouman, "Sentiment analysis of Arabic Jordanian dialect tweets," *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 2, 2019.
[30] A. K. Sangaiah, A. E. Fakhry, M. Abdel-Basset, and I. El-Henawy, "Arabic text clustering using improved clustering algorithms with dimensionality reduction," *Cluster Comput.*, vol. 22, no. S2, pp. 4535–4549, Mar. 2019, doi: 10.1007/s10586-018-2084-4.
[31] N. Bölücü and B. Can, "Unsupervised joint PoS tagging and stemming for agglutinative languages," *ACM Trans. Asian Low-Resource Lang. Inf. Process.*, vol. 18, no. 3, pp. 1–21, Sep. 2019, doi: 10.1145/3292398.
[32] S.-O. Proksch, C. Wratil, and J. Wäckerle, "Testing the validity of automatic speech recognition for political text analysis," *Political Anal.*, vol. 27, no. 3, pp. 339–359, Jul. 2019.
[33] M. Pala, L. Parayitam, and V. Appala, "Unsupervised stemmed text corpus for language modeling and transcription of Telugu broadcast news," *Int. J. Speech Technol.*, vol. 23, no. 3, pp. 695–704, Sep. 2020.
[34] A. Roy, T. Ghorai, K. Ghosh, and S. Ghosh, "Combining local and global word embeddings for microblog stemming," in *Proc. ACM Conf. Inf. Knowl. Manage.*, Nov. 2017, pp. 2267–2270, doi: 10.1145/3132847.3133103.
[35] D. Farrar and J. H. Hayes, "A comparison of stemming techniques in tracing," in *Proc. IEEE/ACM 10th Int. Symp. Softw. Syst. Traceability (SST)*, May 2019, pp. 37–44.
[36] D. Merlini and M. Rossini, "Text categorization with WEKA: A survey," *Mach. Learn. With Appl.*, vol. 4, Jun. 2021, Art. no. 100033.
[37] J. Atwan, M. Wedyan, Q. Bsoul, A. Hammadeen, and R. Alturki, "The use of stemming in the Arabic text and its impact on the accuracy of classification," *Sci. Program.*, vol. 2021, pp. 1–9, Nov. 2021, doi: 10.1155/2021/1367210.
[38] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, Mar. 1980, doi: 10.1108/eb046814.
[39] A. Alnaied, M. Elbendak, and A. Bulbul, "An intelligent use of stemmer and morphology analysis for Arabic information retrieval," *Egyptian Informat. J.*, vol. 21, no. 4, pp. 209–217, Dec. 2020, doi: 10.1016/j.eij.2020.02.004.
[40] P. Koirala and A. Shakya, "A nepali rule based stemmer and its performance on different NLP applications," Feb. 2020, *arXiv:2002.09901*.
[41] M. N. Kassim, S. H. M. Jali, M. A. Maarof, A. Zainal, and A. A. Wahab, "Enhanced text stemmer with noisy text normalization for Malay texts," in *Smart Trends in Computing and Communications*. Birmingham, U.K.: Springer, 2020, pp. 433–444.
[42] B. Azman, "Root identification tool for Arabic verbs," *IEEE Access*, vol. 7, pp. 45866–45871, 2019, doi: 10.1109/ACCESS.2019.2908177.
[43] S. Bessou and M. Touahria, "An accuracy-enhanced stemming algorithm for Arabic information retrieval," 2019, *arXiv:1911.08249*.
[44] M. Ali, S. Khalid, and M. Saleemi, "Comprehensive stemmer for morphologically rich Urdu language," *Int. Arab J. Inf. Technol.*, vol. 16, no. 1, pp. 138–147, 2019.
[45] A. M. Saeed, T. A. Rashid, A. M. Mustafa, R. A. A.-R. Agha, A. S. Shamsaldin, and N. K. Al-Salihi, "An evaluation of reber stemmer with longest match stemmer technique in Kurdish sorani text classification," *Iran J. Comput. Sci.*, vol. 1, no. 2, pp. 99–107, Jun. 2018, doi: 10.1007/s42044-018-0007-4.
[46] A. Jabbar, S. Iqbal, and M. U. G. Khan, "Analysis and development of resources for Urdu text stemming," *Lang. Technol.*, vol. 1, pp. 1–7, Nov. 2016.
[47] M. Ali, S. Khalid, and M. H. Aslam, "Pattern based comprehensive Urdu stemmer and short text classification," *IEEE Access*, vol. 6, pp. 7374–7389, 2018, doi: 10.1109/ACCESS.2017.2787798.
[48] K. Abainia, S. Ouamour, and H. Sayoud, "A novel robust Arabic light stemmer," *J. Exp. Theor. Artif. Intell.*, vol. 29, no. 3, pp. 557–573, May 2017, doi: 10.1080/0952813x.2016.1212100.
[49] P. Selvaramalakshmi, S. H. Ganesh, and J. J. Manoharan, "A novel PSS stemmer for string similarity joins," in *Proc. World Congr. Comput. Commun. Technol. (WCCCT)*, Feb. 2017, pp. 147–150, doi: 10.1109/WCCCT.2016.43.
[50] O. Aldabbas, R. Al-Shalabi, G. Kanan, and M. A. Shehab, "Arabic light stemmer based on regular expression technique," in *Proc. ICSIC*, 2016, pp. 1–9.
[51] N. N. Karanikolas, "Supervised learning for building stemmers," *J. Inf. Sci.*, vol. 41, no. 3, pp. 315–328, Jun. 2015.
[52] S. Sulaiman, K. Omar, N. Omar, M. Z. Murah, and H. A. Rahman, "Jawi stemmer: Evaluation of stemmers based on strength and accuracy," *J. Converg. Inf. Technol.*, vol. 8, no. 12, pp. 19–28, Jul. 2013, doi: 10.4156/jcit.vol8.issue12.3.
[53] M. Kasthuri, S. B. R. Kumar, and S. Khaddaj, "PLIS: Proposed language independent stemmer for information retrieval systems using dynamic programming," in *Proc. World Congr. Comput. Commun. Technol. (WCCCT)*, Feb. 2017, pp. 132–135, doi: 10.1109/WCCCT.2016.39.
[54] W. B. A. Karaa and N. Gribâa, "Information retrieval with porter stemmer: A new version for English," in *Proc. 3rd Int. Conf. Comput. Sci., Eng. Inf. Technol. (CCSEIT)*. Konya, Turkey: KTO Karatay Univ., Jun. 2013, pp. 243–254.
[55] W. B. A. Karaa, "A new stemmer to improve information retrieval," *Int. J. Netw. Secur. Appl.*, vol. 5, no. 4, pp. 143–154, Jul. 2013, doi: 10.5121/ijnsa.2013.5411.
[56] U. Mishra and C. Prakash, "MAULIK: An effective stemmer for Hindi language," *Int. J. Comput. Sci. Eng.*, vol. 4, no. 5, p. 711, 2012.
[57] A. B. Soliman, K. Eissa, and S. R. El-Beltagy, "AraVec: A set of Arabic word embedding models for use in Arabic NLP," *Proc. Comput. Sci.*, vol. 117, pp. 256–265, Jan. 2017, doi: 10.1016/j.procs.2017.10.117.
[58] J. Savoy, "A stemming procedure and stopword list for general French corpora," *J. Amer. Soc. Inf. Sci.*, vol. 50, no. 10, pp. 944–952, 1999, doi: 10.1002/(SICI)1097-4571(1999)50:10<944::AID-ASI9>3.0.CO;2-Q.
[59] J. B. Lovins, "Development of a stemming algorithm," *Mech. Transl. Comput. Linguistics*, vol. 11, nos. 1–2, pp. 22–31, Mar. 1968.
[60] A. Jabbar, S. Iqbal, and M. Ilahi. (2022). *High Performance Stemming Algorithm to Handle Multi-Level Inflections in Urdu Language*. Accessed: Sep. 27, 2023. [Online]. Available: https://www.researchsquare.com/article/rs-1529588/latest
[61] R. Sadia, M. A. Rahman, and M. H. Seddiqui, "N-gram statistical stemmer for Bangla corpus," pp. 2–6, 2019, *arXiv:1912.11612*.
[62] B. P. Pande, P. Tamta, and H. S. Dhami, "Generation, implementation, and appraisal of an N-gram-based stemming algorithm," *Digit. Scholarship Humanities*, vol. 34, no. 3, pp. 558–568, Sep. 2019, doi: 10.1093/llc/fqy053.
[63] J. Dadashkarimi, H. N. Esfahani, H. Faili, and A. Shakery, "SS4MCT: A statistical stemmer for morphologically complex texts," in *Proc. 7th Int. Conf. CLEF Association, CLEF*. Évora, Portugal: Springer, Sep. 2016, pp. 201–207.
[64] T. Brychcín and M. Konopík, "HPS: High precision stemmer," *Inf. Process. Manage.*, vol. 51, no. 1, pp. 68–91, Jan. 2015, doi: 10.1016/j.ipm.2014.08.006.
[65] S. Ferilli, F. Esposito, and D. Grieco, "Automatic learning of linguistic resources for stopword removal and stemming from text," *Proc. Comput. Sci.*, vol. 38, pp. 116–123, Jan. 2014.
[66] B. P. Pande, P. Tamta, and H. S. Dhami, "Generation, implementation and appraisal of an N-gram based stemming algorithm," 2013, *arXiv:1312.4824*.
[67] M. S. Husain, "An unsupervised approach to develop IR system : The case of Urdu," *Int. J. Artif. Intell. Appl.*, vol. 4, no. 5, pp. 77–87, Sep. 2013, doi: 10.5121/ijaia.2013.4506.
[68] A. K. Pandey and T. J. Siddiqui, "An unsupervised Hindi stemmer with heuristic improvements," in *Proc. 2nd Workshop Anal. Noisy Unstructured Text Data*, Jul. 2008, pp. 99–105.
[69] F. Ahmed and A. Nürnberger, "Evaluation of n-gram conflation approaches for Arabic text retrieval," *J. Amer. Soc. Inf. Sci. Technol.*, vol. 60, no. 7, pp. 1448–1465, Jul. 2009.

[70] R. Al-Shalabi, G. Kanaan, S. Ghwanmeh, and F. M. Nour, "Stemmer algorithm for Arabic words based on excessive letter locations," in *Proc. Innov. Inf. Technol. (IIT)*, Nov. 2007, pp. 456–460.

[71] M. Bacchin, N. Ferro, and M. Melucci, "A probabilistic model for stemmer generation," *Inf. Process. Manage.*, vol. 41, no. 1, pp. 121–137, Jan. 2005, doi: 10.1016/j.ipm.2004.04.006.

[72] P. McNamee, C. Nicholas, and J. Mayfield, "Addressing morphological variation in alphabetic languages," in *Proc. 32nd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr. (SIGIR)*, Jul. 2009, pp. 75–82, doi: 10.1145/1571941.1571957.

[73] F. S. Alotaibi and V. Gupta, "A cognitive inspired unsupervised language-independent text stemmer for information retrieval," *Cognit. Syst. Res.*, vol. 52, pp. 291–300, Dec. 2018, doi: 10.1016/j.cogsys.2018.07.003.

[74] J. Singh and V. Gupta, "An efficient corpus-based stemmer," *Cognit. Comput.*, vol. 9, no. 5, pp. 671–688, Oct. 2017, doi: 10.1007/s12559-017-9479-z.

[75] I. Boukhalfa, S. Mostefai, and N. Chekkai, "A study of graph based stemmer in Arabic extrinsic plagiarism detection," in *Proc. 2nd Medit. Conf. Pattern Recognit. Artif. Intell.*, Mar. 2018, pp. 27–32.

[76] J. H. Paik, S. K. Parui, D. Pal, and S. E. Robertson, "Effective and robust query-based stemming," *ACM Trans. Inf. Syst.*, vol. 31, no. 4, pp. 1–29, Nov. 2013, doi: 10.1145/2536736.2536738.

[77] J. H. Paik and S. K. Parui, "A fast corpus-based stemmer," *ACM Trans. Asian Lang. Inf. Process.*, vol. 10, no. 2, pp. 1–16, Jun. 2011, doi: 10.1145/1967293.1967295.

[78] A. Zitouni, A. Damankesh, F. Barakati, M. Atari, M. Watfa, and F. Oroumchian, "Corpus-based Arabic stemming using N-grams," in *Proc. 6th Asia Inf. Retr. Societies Conf. (AIRS)*. Taipei, Taiwan: Springer, Dec. 2010, pp. 280–289.

[79] P. Majumder, M. Mitra, S. K. Parui, G. Kole, P. Mitra, and K. Datta, "YASS: Yet another suffix stripper," *ACM Trans. Inf. Syst.*, vol. 25, no. 4, p. 18, Oct. 2007, doi: 10.1145/1281485.1281489.

[80] T. Korenius, J. Laurikkala, K. Järvelin, and M. Juhola, "Stemming and lemmatization in the clustering of Finnish text documents," in *Proc. 13th ACM Int. Conf. Inf. Knowl. Manage.*, Nov. 2004, pp. 625–633.

[81] N. Bouhriz, F. Benabbou, and H. Benlahmer, "Text concepts extraction based on Arabic WordNet and formal concept analysis," *Int. J. Comput. Appl.*, vol. 111, no. 16, pp. 30–34, Feb. 2015.

[82] M. Basu, A. Roy, K. Ghosh, S. Bandyopadhyay, and S. Ghosh, "A novel word embedding based stemming approach for microblog retrieval during disasters," in *Proc. 39th Eur. Conf. IR Res. (ECIR)*. Aberdeen, U.K.: Springer, Apr. 2017, pp. 589–597.

[83] M. Mezzi, N. Benblidia, and X. Huang, "A semantically enriched context-aware stemming algorithm," *J. Integr. Design Process Sci.*, vol. 21, no. 4, pp. 5–24, Oct. 2018.

[84] K. K. Agbele, A. O. Adesina, N. A. Azeez, and A. P. Abidoye, "Context-aware stemming algorithm for semantically related root words," *Afr. J. Comput. ICTs*, vol. 5, no. 4, pp. 33–42, Jun. 2012.

[85] K. Boukhari and M. N. Omri, "RAID: Robust algorithm for stemming text document," *Int. J. Comput. Inf. Syst. Ind. Manag. Appl.*, vol. 8, no. 1, pp. 235–246, 2016.

[86] G. Adam, K. Asimakis, C. Bouras, and V. Poulopoulos, "An efficient mechanism for stemming and tagging: The case of Greek language," in *Proc. Int. Conf. Knowl.-Based Intell. Inf. Eng. Syst.*, in Lecture Notes in Computer Science: Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics, vol. 6278, 2010, pp. 389–397, doi: 10.1007/978-3-642-15393-8_44.

[87] E. T. Al-Shammari and J. Lin, "Towards an error-free Arabic stemming," in *Proc. 2nd ACM Workshop Improving English Web Searching*, Oct. 2008, pp. 9–16.

[88] M. Hadni, S. A. Ouatik, and A. Lachkar, "Effective Arabic stemmer based hybrid approach for Arabic text categorization," *Int. J. Data Mining Knowl. Manage. Process*, vol. 3, no. 4, pp. 1–14, Jul. 2013, doi: 10.5121/ijdkp.2013.3401.

[89] S. Khoja and R. Garside, "Stemming Arabic text," Dept. Comput. Sci., Lancaster Univ., Lancaster, U.K., Tech. Rep., 1999.

[90] N. Saharia, U. Sharma, and J. Kalita, "Stemming resource-poor Indian languages," *ACM Trans. Asian Lang. Inf. Process.*, vol. 13, no. 3, pp. 1–26, Oct. 2014.

[91] R. S. Patil and S. R. Kolhe, "Inflectional and derivational hybrid stemmer for sentiment analysis: A case study with Marathi tweets," in *Proc. 4th Int. Conf. Recent Trends Image Process. Pattern Recognit. (RTIPR)*. Msida, Malta: Springer, 2022, pp. 263–279.

[92] B. Mahalingam, S. Kannan, and V. Gurusamy, "Bruteporter: A hybrid approach," *Int. J. Educ. Manage. Eng.*, vol. 8, no. 5, pp. 11–17, Sep. 2018.

[93] R. Rouibia, I. Belhadj, and M. A. Cheragui, "JIDR: Towards building hybrid Arabic stemmer," in *Proc. Int. Conf. Math. Inf. Technol. (ICMIT)*, Dec. 2017, pp. 183–190, doi: 10.1109/MATHIT.2017.8259714.

[94] H. Taghi-Zadeh, M. H. Sadreddini, M. H. Diyanati, and A. H. Rasekh, "A new hybrid stemming method for Persian language," *Digit. Scholarship Humanities*, vol. 32, no. 1, pp. 209–221, 2017, doi: 10.1093/llc/fqv053.

[95] F. Momenipour and M. R. Keyvanpour, "PHMM: Stemming on Persian texts using statistical stemmer based on hidden Markov model," *Int. J. Inf. Sci. Manag.*, vol. 14, no. 2, pp. 107–117, 2016.

[96] C. Sitaula, "A hybrid algorithm for stemming of Nepali text," *Intell. Inf. Manag.*, vol. 5, no. 4, p. 139, 2013, Art. no. 34712.

[97] C. D. Paice, "Another stemmer," *ACM SIGIR Forum*, vol. 24, no. 3, pp. 56–61, Nov. 1990, doi: 10.1145/101306.101310.

[98] W. B. Frakes and C. J. Fox, "Strength and similarity of affix removal stemming algorithms," *ACM SIGIR Forum*, vol. 37, no. 1, pp. 26–30, Apr. 2003.

[99] S. R. Sirsat, V. Chavan, and H. S. Mahalle, "Strength and accuracy analysis of affix removal stemming algorithms," *Int. J. Comput. Sci. Inf. Technol.*, vol. 4, no. 2, pp. 265–269, 2013.

[100] H. A. Almuzaini and A. M. Azmi, "Impact of stemming and word embedding on deep learning-based Arabic text categorization," *IEEE Access*, vol. 8, pp. 127913–127928, 2020, doi: 10.1109/ACCESS.2020.3009217.

[101] B. Nathani, N. Joshi, and G. N. Purohit, "Design and development of unsupervised Stemmer for Sindhi language," *Proc. Comput. Sci.*, vol. 167, pp. 1920–1927, Jan. 2020.

[102] H. Kaur and P. K. Buttar, "A rule-based stemmer for Punjabi adjectives," *Int. J. Adv. Res. Comput. Sci.*, vol. 11, no. 6, pp. 15–19, Dec. 2020, doi: 10.26483/ijarcs.v11i6.6665.

[103] A. A. Suryani, D. H. Widyantoro, A. Purwarianti, and Y. Sudaryat, "The rule-based sundanese stemmer," *ACM Trans. Asian Low-Resource Lang. Inf. Process.*, vol. 17, no. 4, pp. 1–28, Dec. 2018, doi: 10.1145/3195634.

[104] S. B. Rodzman, M. F. I. A. Ronie, N. K. Ismail, N. A. Rahman, F. Ahmad, and Z. M. Nor, "Analyzing Malay stemmer performance towards fuzzy logic ranking function on Malay text corpus," in *Proc. 4th Int. Conf. Inf. Retr. Knowl. Manage. (CAMP)*, Mar. 2018, pp. 1–6.

[105] D. S. Maylawati, W. B. Zulfikar, C. Slamet, M. A. Ramdhani, and Y. A. Gerhana, "An improved of stemming algorithm for mining Indonesian text with slang on social media," in *Proc. 6th Int. Conf. Cyber IT Service Manage. (CITSM)*, Aug. 2018, pp. 1–6.

[106] H. B. Patil and A. S. Patil, "MarS: A rule-based stemmer for morphologically rich language Marathi," in *Proc. Int. Conf. Comput., Commun. Electron. (Comptelix)*, Jul. 2017, pp. 580–584, doi: 10.1109/COMPTELIX.2017.8004036.

[107] N. Desai and B. Dalwadi, "An affix removal stemmer for Gujarati text," in *Proc. 3rd Int. Conf. Comput. Sustain. Global Develop. (INDIACom)*, Mar. 2016, pp. 2296–2299.

[108] H. Sever and Y. Bitirim, "FindStem: Analysis and evaluation of a Turkish stemming algorithm," in *Proc. Int. Symp. String Process. Inf. Retr.* Pisa, Italy: Springer, 2003, pp. 238–251.

[109] J. Singh and V. Gupta, "A novel unsupervised corpus-based stemming technique using lexicon and corpus statistics," *Knowl.-Based Syst.*, vol. 180, pp. 147–162, Sep. 2019, doi: 10.1016/j.knosys.2019.05.025.

[110] J. H. Paik, "A novel TF-IDF weighting scheme for effective ranking," in *Proc. 36th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2013, pp. 343–352, doi: 10.1145/2484028.2484070.

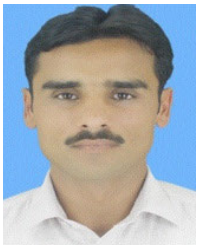
[111] J. H. Paik, M. Mitra, S. K. Parui, and K. Järvelin, "GRAS: An effective and efficient stemming algorithm for information retrieval," *ACM Trans. Inf. Syst.*, vol. 29, no. 4, pp. 1–24, Dec. 2011, doi: 10.1145/2037661.2037664.

[112] A. Mateen, M. K. Malik, Z. Nawaz, H. M. Danish, M. H. Siddiqui, and Q. Abbas, "A hybrid stemmer of Punjabi Shahmukhi script," *Int. J. Comput. Sci. Netw. Secur.*, vol. 17, no. 8, pp. 90–97, 2017.

[113] Y. A. Al-Lahham, "Index term selection heuristics for Arabic text retrieval," *Arabian J. Sci. Eng.*, vol. 46, no. 4, pp. 3345–3355, Apr. 2021, doi: 10.1007/s13369-020-05022-3.

[114] G. Müller, J. Englisch, and A. Opitz, "Extraction from NP, frequency, and minimalist gradient harmonic grammar," *Linguistics*, vol. 60, no. 5, pp. 1619–1662, Sep. 2022.

[115] S. M. J. Rizvi and M. Hussain, "Analysis, design and implementation of Urdu morphological analyzer," in *Proc. Student Conf. Eng. Sci. Technol.*, Aug. 2005, pp. 7374–7389, doi: 10.1109/sconest.2005.4382901.
[116] R. Medjedoub, "A comparison of English and Arabic noun inflectional morphology," *Hum. Sci. J.*, vol. 33, pp. 9–22, Dec. 2022.
[117] V. M. Orengo and C. R. Huyck, "A stemming algorithmm for the Portuguese language," in *Proc. SPIRE*, 2001, pp. 186–193.
[118] M. Akasereh, "Multilingual and domain-specific IR: A case study in cultural heritage," Unpublished thesis, Univ. Neuchâtel, Neuchâtel, Switzerland, 2015.
[119] T. Fatima, R. U. Islam, M. W. Anwar, M. H. Jamal, M. T. Chaudhry, and Z. Gillani, "STEMUR: An automated word conflation algorithm for the Urdu language," *ACM Trans. Asian Low-Resource Lang. Inf. Process.*, vol. 21, no. 2, pp. 1–20, Mar. 2022, doi: 10.1145/3476226.
[120] E. Mohamed, S. Elmougy, O. A. S. Ibrahim, and M. Aref, "Semantic relatedness based query translation disambiguation approach for cross-language web search," *J. King Saud Univ.—Comput. Inf. Sci.*, vol. 29, no. 2, pp. 164–170, 2017.
[121] Y. Jaafar, D. Namly, K. Bouzoubaa, and A. Yousfi, "Enhancing Arabic stemming process using resources and benchmarking tools," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 29, no. 2, pp. 164–170, Apr. 2017, doi: 10.1016/j.jksuci.2016.11.010.



ABDUL JABBAR received the master's degree in computer science from the Department of Computer Science, Institute of Southern Punjab, Multan. He is currently pursuing the Ph.D. degree with the Department of Computer Science, COMSATS University Islamabad, Pakistan. His research interests include natural language processing, computational linguistics, text mining, and artificial intelligence.

SAJID IQBAL received the Ph.D. degree from the Department of computer Science, University of Engineering and Technology, Lahore, Pakistan. Currently, he is an Assistant Professor with the Department of Information Systems, College of Computer Sciences and Information Technology, King Faisal University, Al-Ahsa, Saudi Arabia. He has published more than 30 papers in local and international journals and conferences. His research interests include medical image analysis, natural language processing, and computer vision.

MANZOOR ILAHI TAMIMY received the master's degree in computer science from Gomal University, Dera Ismail Khan, Pakistan, in 1998, under the supervision of D. I. Khan, and the Ph.D. degree in computer science. He rejoined the Department of Computer Science, CIIT, as an Assistant Professor, in 2009. Currently, he is a Professor with the Department of Computer Science. In 2005, he was awarded the CIIT Scholarship for the Ph.D. Studies with GSCAS, Beijing, China.

AMJAD REHMAN (Senior Member, IEEE) received the Ph.D. degree from the Faculty of Computing, Universiti Teknologi Malaysia (UTM), Malaysia, specializing in information security using image processing techniques, in 2010. He is currently an Associate Professor with CCIS, Prince Sultan University, Riyadh, Saudi Arabia. He is also a PI in several projects and completed projects funded by MoHE Malaysia, Saudi Arabia. His research interests include bioinformatics, the IoT, information security, and pattern recognition. He received the Rector Award for the 2010 Best Student from UTM.

SAEED ALI BAHAJ received the Ph.D. degree from Pune University, India, in 2006. He is currently an Associate Professor with the Department of Computer Engineering, Hadramout University, and the MIS Department, COBA, Prince Sattam bin Abdulaziz University, Al-Kharj, Saudi Arabia. His main research interests include information management, forecasting, information engineering, and information security.

TANZILA SABA (Senior Member, IEEE) received the Ph.D. degree in document information security and management from the Faculty of Computing, Universiti Teknologi Malaysia (UTM), Malaysia, in 2012. Currently, she is a Full Professor with the College of Computer and Information Sciences, Prince Sultan University (PSU), Riyadh, Saudi Arabia, and also the AIDA Lab Leader. She has published over 300 publications in high-ranked journals. Her primary research interests include bioinformatics, data mining, and classification using AI models. She was awarded the Best Research of the Year Award from PSU, from 2013 to 2016. Due to her excellent research achievement, she is included in Marquis Who's Who (S&T), in 2012. She is an editor of several reputed journals and on a panel of TPC of international conferences. She won the Best Student Award from the Faculty of Computing, UTM, in 2012.