PWMStem: A Corpus-Based Suffix Identification and Stripping Algorithm for Multi-lingual Stemming

Abdul Jabbar [1], Manzoor Illahi [1], Sajid Iqbal [2], Amjad Rehman Khan [3, *], Narmine ElHakim [3], and Tanzila Saba [3]

[1] Department of Computer Science, Comsats University Islamabad (CUI), Main Campus, Park Road, Tarlai Kalan, Islamabad 45550, Pakistan; Email: a.jabbar73@hotmail.com (A.J.), tamimy@comsats.edu.pk (M.I.)
[2] Department of Information Systems, College of Computer Science and Information Technology, King Faisal University, Saudi Arabia; Email: siqbal@kfu.edu.sa (S.I.)
[3] Artificial Intelligence and Data Analytics Lab CCIS Prince Sultan University Riyadh 11586, Saudi Arabia; Email: nhakim@psu.edu.sa (N.E.), tsaba@psu.edu.sa (T.S.)
*Correspondence: arkhan@psu.edu.sa (A.R.K.)

Abstract—**Stemming is a common preprocessing method aggregating all word variants to a standard stem to aid various Natural Language Processing (NLP) tasks. This work proposes a new unsupervised corpus-based stemmer that identifies the candidate suffixes using pivot word matching. Then candidate suffix statistics are used to remove the potential suffixes. After this, lexical similarity is measured to cluster the morphological related words. Finally, the smallest word in each cluster is designated as a stem. To quantify the performance of proposed method, two corpus-based and two linguistic knowledge-based stemmers for Urdu and English languages are used. The performance of each stemmer is evaluated on two different datasets for each language. The results show that the proposed PWMStem method outperforms the selected stemmers, achieving an accuracy of 0.876 for Urdu and 0.877 for English. To assess the performance of PWMStem through different aspects multiple evaluation metrics are used. The evaluation scores of other metrics are Index Compression Factor (ICF) = 73, Mean Number of Words per Conflation Class (MWC) = 3.7 for Urdu, and ICF = 71 and MWC = 3.5 for English. In the Urdu dataset, PWMStem achieved the lowest Under-stemming Index (UI) of 0.026479, Over-stemming Index (OI) of 0.000021, and an Error Rate Relative to Truncation (ERRT) of 0.610. In the English dataset, the values for UI, OI, and ERRT were measured as 0.102089, 0.000015, and 0.498, respectively.**

Keywords—**corpus-based stemming, morphology, natural language processing, Urdu stemmer, words inflection**

I. Introduction

With the advent of the Internet, the generation and use of natural language content in various languages has increased exponentially. Several methods, known as pre-processing methods, are used to transform data into a desired form before applying a task-specific algorithm to extract the required information. Among these, stemming is an prominent method that aggregates all variations of a word into one morphologically related class to aid natural language processing tasks [1].

Urdu is different from other languages like English in terms of its linguistic, and phonetic rules. Urdu script is written from right-to-left direction. It is highly Persianised/Arabicised. It is the national language of Pakistan as well as spoken in various parts of world like United Arab Emirates (UAE), United Kingdom (UK), United States (US) and in many parts of India[1].

Many language processing tools are available for the English language. NLP tools for Urdu still need to be improved in number and it is required to build more efficient tools. New tools are always required to build as new tasks and applications are emerging with time.

Stemming serves multiple roles in language processing which include reducing the size of the index file, reducing the number of features for classification and lexical transfer learning tasks [2]. Stemming can be utilized as a crucial preprocessing tool in various Natural Language Processing (NLP) tasks like Text Classification (TC), automatic indexing, lexical analysis, information extraction, and text summarization [3]. It also resolves the query mismatch problem in Information Retrieval (IR) systems, improving the system's recall. The emergence of new applications of NLP needs more sophisticated and high-performance tools like stemming.

Different approaches have been proposed to develop stemmers, including rule-based, supervised, semi-supervised, and unsupervised machine learning methods [4]. In this work, we have proposed a new unsupervised corpus-based method that identifies candidate suffixes using Pivot Word Matching (PWM) and removes potential suffixes based on their statistics, then forms clusters of words with lexical similarity and

Manuscript received April 24, 2023; revised June 14, 2023; accepted July 3, 2023; published August 28, 2023.

[1] https://www.britannica.com/topic/Urdu-language

designates the smallest word in each cluster as the stem. We evaluate the PWMStem for Urdu and the English corpus, and the results show that it is an efficient approach for both languages. Following list presents the contributions of the work:

- **A new Unsupervised Algorithm**: PWMstem is an unsupervised corpus-based stemmer developed considering Urdu words' structure.
- **Language-independent Approach**: PWMstem cognitively learns morphological patterns without predefined linguistic rules of any language, making PWMstem a language-independent method.
- **Multi-level Morphology Handling**: In contrast to other unsupervised stemmers, PWMstem can handle morphological words, which may consist of multi-level suffixes/long suffixes.
- **Dataset Development**: A custom dataset is developed for the Urdu language to train and evaluate the proposed method.
- **Improved Performance**: PWMstem shows improved performance over state-of-the-art stemmers.

The rest of the article is organized as follows: the review of existing works and comparison is given in Section II. Section III depicts a detailed description of the PWMStem algorithm. Section IV portrays the evaluation results of the proposed stemmer. Result and discussion present in Section V. Finally, Section VI provides the conclusion and future work directions.

II. Literature Review

Various stemmers of different flavors and natures have been proposed in the literature. Regarding design and functions, stemmers may be categorized into two major classes: language-dependent and independent [5]. Language-dependent stemmers use specific re-characterized language-associated requirements to represent the morphological variations of the words. These language-related rules are manually developed by language experts [6]. Several language-dependent stemmers have been designed for English and other languages. The literature review shows that multiple stemmers for minor languages have also been developed. Creating a language-dependent stemmer requires additional assets, tools, and relevant expertise. On the other hand, language-independent and statistical methods are used to obtain linguistic features, and a stem is extracted via these features. In the following subsections, we review relevant linguistic knowledge-based and language-independent stemmers.

A. Language Dependent Stemmers

Aba *et al.* [7] proposed a linguistic-based stemming algorithm for Urdu, which utilizes a predefined prefix and suffix list to recognize the affix part in a query word. Further, infixes are recognized with specific letters positioned in the query words, and appropriate infixes rules are applied. The study identifies some exceptional cases, such as سجود [prostration] and نقوش [Impressions], that have similar patterns, and such exceptional cases are treated differently to obtain stem. Alshalabi *et al.* [8] developed an Arabic linguistic-based stemmer that utilizes one character's prefix and suffix and ignores the higher-length affixes. The internal structure of the singular word changed when converting to plural form know as broken/irregular plural word such as [children] from singular [child], in which singular form [child] suffers internal changes by adding the [alif] and [alif laam] to plural form [children]. whereas the singular form of words is not broken when making their plural without changing the internal structure, such as [female teachers] from singular [female teacher]. In this case, the internal structure of the singular word did not change, and the plural is formed by adding the اalif tee] at the end. Various patterns of length 4–6 is designed to get the stem of broken plural words. Three patterns are designed for the words of length four, 15 patterns for length five, and six patterns for words length six. Alshalabi *et al.* [9] refined "The Information Science Research Institute's (ISRI) Arabic stemmer", which identifies the affixes of up to five characters which are then removed according to word length instead of the pattern. Alnaied *et al.* [5] developed a list of rules to produce the stem consisting of three phases: substring tagging, rule matching, and anti-rule matching. They defined 58 prefixes, three infixes' letters, and 25 suffixes. Saeed *et al.* [10] designed an iterative stemmer for the Persian language using prefix and suffix lists to find the prefixes and suffixes to remove them iteratively, resulting in improved performance for the classification task. Harouni *et al.* [11] designed a stemmer for the Sundanese language that iteratively removes the longer affixes before the shorter ones to obtain the stem.

Jabbar *et al.* [12] presented a multi-step Urdu stemmer that was evaluated on the custom-designed text and word corpus. The stemmer operates in multiple phases, including affix striping, template matching, and table look-up. Khan *et al.* [13] suggested an Urdu linguistic rule-based stemmer that identifies the affix letters by the predefined pattern, affix list, and exceptional list to derive the stem. The porter stemmer [14] is the most famous English stemmer widely used in IR systems. It works in five steps using rules and conditions matching vowel and consonant pattern sequences. Porter stemmer [14] in the first step, handles inflectional suffixes. In the next three steps, it handles the derivational suffixes and performs the recoding. Lovins is another language-dependent stemmer [15]. It comprises 294 suffixes associated with 29 conditions that determine the eligibility of a suffix for its removal. It also contains 35 transformation rules based on the longest match criterion. After removing suffixes, the recoding rules are applied to convert the stem to a linguistically correct word.

B. Language-Independent Stemmers

Singh and Gupta [16] developed a statistical stemmer using linguistic, co-occurrence similarity and suffix pair frequency to compute the morphological correlation among words to form a cluster. The common prefix in the cluster is then retrieved as a stem. Alotaibi and Gupta [4] proposed a language-independent stemmer that uses the

Jaccard distance similarity measure to cluster morphologically related words. The threshold of the similarity metrics is set to 1.5 by authors. Kasthuri *et al.* [17] also designed a language-independent stemmer using Levenshtein Edit Distance (LED) and the Longest Common Subsequence (LCS) metrics to find the similarity between the strings. The procedure first groups the words based on the length of the common prefix and then selects the most common word as a stem using filtering rules. The acceptable filtering rule is defined as LED < LCS. Chavula and Suleman [18] showed that the orthographic similarity measure did not indicate the morphological distribution of morphemes. Subsequently, they proposed a weighted similarity measure that uses Ordered Weighted Aggregate (OWA) to conflate morphologically similar words in the corpus using common letter patterns.

Singh and Gupta [19] developed an efficient stemmer that used the Jaro-Winkler distance with a variant to cluster morphologically different words using the graph-based clustering method. Brychcín and Konopík [20] designed a stemming technique that considerably improves precision at the cost of a slight decrease in recall. The method uses each lexical and co-occurrence metric to learn morphological rules for stemming. Husain *et al.* [21] constructed a language-independent stemmer using the n-gram technique. The system produces n-gram tokens, and then the suffix list is generated from these tokens. The extracted suffixes are eliminated using criteria of frequency and length of the suffixes. The authors claimed that frequency-based criteria give better results than the suffix length-based method. Paik *et al.* [22] developed an unsupervised stemming algorithm that discovers suffixes based on their frequency. They divide the corpus into groups using the average word length as prefix matching. Then, the potential suffix and longest common prefix feature to formulate the morphologically related word classes. Paik and Parui [23] extracted the suffixes using the co-occurrence of the trailing part of the words in the corpus and grouped the morphologically related words on the bases of suffix frequency and prefix matching criteria.

Majumder *et al.* [24] recommended as a method to cluster morphologically related words using four similarity metrics to cluster morphologically related words. Goldsmith [25, 26] proposed an unsupervised stemming model which produces all possible pairs of stems and suffixes known as signatures, then Mutual Information (MI) filters out the stem and suffix. The minimum description length is used as the threshold value of Lee abd Goldsmith [27] built a software system called Linguistica 5 utilizing the framework of [25, 26] for stemmers evaluation which is also used in this study.

C. Comparison and Evaluation

In this Section we analyze the various features of the state-of-the-art stemmer. The comparison is presented in Table I.

TABLE I. COMPARISON OF CHARACTERISTICS OF THE STEMMERS

Features	Ref.
• Initial classes were created using the average word length as a common prefix. According to Peter Norvig[2], the average word length for the English language is five. • Peter Norvig’s corpus analysis shows that most English words have lengths of three and four letters, which are ignored. • Suffixes are obtained by matching common prefixes and the potential suffixes are filtered using frequency of co-occurrence of suffixes. They typically strip only the last suffix of the query word. In the English word like ‘helpfulness’ would be striped to ‘helpful’ by removing ‘ness’ as suffix by this method. To cut off the whole suffix part (fulness) is a harder task. • Features extracted to create the morphologically related words class are: the lexical similarity, the co-occurrence frequency of the words and potential suffix is used. • This method produces the common part as stem for instance morphologically related word class is (share, shares, shared, sharing) and return the ‘shar’ as stem which may improve the performance of information retrieval but may not be beneficial for machine translation and speech recognition. • Language independent in nature.	Singh and Gupta [16]
• Initial classes were created using the average words length as common prefix. According to Peter Norvig, the average word length for the English language is five. • The words with smaller word length are ignored. • Suffixes are not retrieved. • The morphologically related words are clustered using lexical similarity. In such methods, longest suffixes/multilevel suffixes are not properly removed. Therefore, the system suffers with under stemming errors. • The method is application oriented as the common part of the morphological related cluster is considered the stem. • Language independent in nature.	Mujeeb *et al.* [4]
• Segment the word in possible stem suffix pair. • Mutual Information of the stem suffix pair is used to assess the correct stem and suffix. • This method ignored the multilevel suffixes such as the word ‘admirers’. • A lot of unnecessary tokens are created which increase the computational cost and space on disk. • Language independent in nature.	Goldsmith [25, 26]

[2] http://norvig.com/mayzner.html

• Use the predefined suffixes list to recognize and remove suffixes. • After stripping the suffix, the recoding rules are applied to convert the extracted stem into a valid stem. However, in some cases, it produces an invalid word such as 'stay' stems to 'stai'. • Due to minimal length-based nature, the method suffers the under stemming errors. • The method is specifically developed for English words	Porter [14]
• This method uses the predefined suffixes list to recognize and remove them. • The suffix list is short of many suffixes. • The minimum length of stem is two letters which significantly produce over stemming errors. • Consider only English words	Lovins [15]
• Used predefine affix list to identify and remove the suffixes. • The affix list is incomplete. • Unable to treat the new affixes which are not included in the predefined affix list. • English loan words did not stem correctly. • Hybrid words are not handled. • The method is language dependent	Rehman and Saba [28]
• Uses predefine affix list to identify and remove the suffixes. • Unable to treat the new affixes which are not included in the predefined affix list. • Hybrid words are not handled. • The method is language dependent	Jabbar *et al*. [12]
• Initial corpus is divided into the various clusters based on the first three letters as common in the case of Urdu as well as English language. • The suffix part is segmented from the query word on the bases of Pivot Word (PW) matching instead of common letters matching. PW based obtained suffixes usually linguistically true. However, we further filter the potential suffixes which have certain threshold value. • In contrast to prior studies, the Lexicon similarity measures after removing the suffix are used. In this way, the long or multilevel suffix words are also allocated the correct morphological cluster. • Efficiently handle the English loan words, which written in Urdu script. • Hybrid words are also handled. • The smallest word from these morphological clusters is considered the stem of all the word in that cluster.	Proposed

III. The Proposed Stemming Method

The main objective of this work is to design a novel and effective corpus-based stemming technique that can serve as a universal tool in various NLP applications. The proposed technique groups morphologically related words appearing in the corpus using lexical similarity and suffix statistics. This algorithm acquires unique words as input from the corpus and produces a set of morphologically related words. Our algorithm works in three phases, as depicted in Fig. 1.

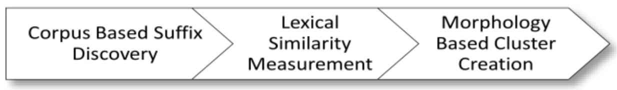
Corpus Based Suffix Discovery
Lexical Similarity Measurement
Morphology Based Cluster Creation

Figure 1. Phases of PWMStem method.

TABLE II. Description of Symbols Used in Methodology

Symbol	Meaning
s_i	Suffix with index i
w_i	A word with index i
w_j	A word with j
ps	A word p with attached suffix s
qs	A word q with attached suffix s
ws_i	Word with suffix s and index i
r_{pq}	Similarity score between word p and q
C_i	Class with index i
C_m	Class of morphologically related words
PW_i	Pivot Word (PW) with index i

Table II lists the symbols which are used in the following discussion.

A. Corpus Development

For the development and evaluation of the PWMStem algorithm, two corpora of English and Urdu languages are used. These include Open American National Corpus (OANC)[3] and English Lemma Dataset compiled by Lin Wei[4], the first dataset for the Urdu language is taken from [29] and the second dataset is a custom-made named as URSTEM, which contains Urdu morphological and related words. The sources of URSTEM include grammar books [30–33] on morphology [34] and an online resource provided by the Center for Language Engineering, University of Engineering and Technology, Lahore, Pakistan[5]. The preprocessing methods include the removal of punctuation, numbers, non-Urdu characters, and Urdu diacritics. After the preprocessing, stop words and duplications were removed from the corpus. Then the dataset adaptation is grouped in the form of morphologically related words. This step required deep consideration and was time-consuming. Multiple corpora are used to evaluate the designed method to ensure a language-independent assessment. An overview of the datasets used in the study is given in Table III.

Paice's [35, 36] evaluation method require a list of morphologically related groups of words. Following the criteria designed by Paice [35, 36] we have categorized the corpus into morphological groups or clusters. This helps to

[3] https://anc.org/SecondRelease/data/ANC-all-lemma.txt
[4] https://github.com/skywind3000
[5] http://oud.cle.org.pk/

compute the performance scores of the proposed method. Each cluster has the following two properties:

- According to Paice [35, 36] morphologically related words must share a common stem. The developed corpus has a minimum of three letters length for common stem.
- Each cluster has a minimum of two words. This condition is also formulated by Paice [35, 36]. An example of morphologically related word forms from the Urdu dataset URSTEM and the English Lemma data set is mentioned in dataset.

TABLE III. DATASET DESCRIPTIONS

Dataset	Lang.	Word Count	Distinct words	Clusters Count	Avg. Cluster Size
OANC	English	221,64985	23,210	11,245	2.1
English Lemma	English	186,523	33,695	8,982	3.8
Humayoun	Urdu	152,264	8,309	2,384	3.5
URSTEM	Urdu	67,126	67,126	19,652	3.4

B. Corpus-Based Suffix Discovery

The morphological variations in most inflectional languages are formed through suffixation, which is the addition of the suffix to the root word to form a new word [37, 38]. Although the stem may be present in any part of query words, this study does not deal with such cases. We assume that a suffix is present in the trailing part of a word or its inflectional form; hence the notation ws is used where w is the stem and s is the suffix. We start with a list of words, such as $\{ws_1, ws_2, \dots, ws_n\}$ is produced by a Pivot Word (PW) from the corpus, then the suffixes $s_1, s_2, \dots, s_n$ are retrieved. The generated suffixes may not be linguistically correct. Statistics like suffix frequency and length are then used to filter valid suffixes.

Using Eq. (1), a Pivot Word (PW) is selected for each cluster, and based on the PW, Candidate Suffixes (CS) of length $(n = 1,2, \dots)$ are extracted, and then Pivot Word Matching (PWM) is performed using Eq. (2). After this, CS suffixes are extracted using Eq. (3). The complete procedure of discovering CS suffixes is described Algorithm 1.

Algorithm 1: Discovering suffixes

```
1   Input: Unannotated Corpus (UC)
2   Construct the Unique Words List (UWL) after
    preprocessing and normalization
3   Arrange the UWL in alphabetical order
4   for each word W in UWL do
5     Construct $C_i = \{w_1, w_2, w_3, \dots, w_n\}$ using Eq.(2)
    : $w_i$ is variant forms of word W
    end for
6   for each cluster Ci in cluster list C
7     for each item $W_j$ in $C_i$ do
8         the suffix is extracted from $W_j$ by Eq.3 and
    added to the CS list
9     end for
10  end for
```

The example of top 20 CS of English depicted in Figs. 2 and 3, top 20 CS of Urdu are shown. Potential Suffixes (PS) are found from CS using suffix length and frequency criteria. The threshold value of suffix length and frequency are determined experimentally. Let w_i and w_j are two strings sharing a common prefix and w_i is selected as a PW word and w_j is the variant form of the w_i, as given in Eq. (1).

$$len(w_i) \le len(w_j) \tag{1}$$

where $(i < j) \wedge (w_i = PW) \wedge \; w_j = variant(PW)$.

Let w_i is a PW then:

$$PWM(w_i, w_j) = \frac{|S(w_i, w_j)|}{|w_i|} \tag{2}$$

where $S(w_i, w_j)$ is the common part between w_i and w_j starting from index zero to $len(w_i)$. It is to note that the zero index will differ for Urdu and English languages due to their orientation. The suffix is extracted by Eq. (3).

$$suff(W) = W^{Z}_{PWM(w_i, w_j)} \tag{3}$$

where $PWM(w_i, w_j) = min\left(LCCP(S_{i,}S_{j,})\right) + 1$ and Z is a size of string w_j.

Following Table IV lists few examples of candidate suffixes.

TABLE IV. EXAMPLE OF CANDIDATE SUFFIXES IN ENGLISH

English Corpus	
affect, affectation, affectations, affected, affectedly, affecteth, affecting, affection, affectionate, affectionately, affections, affects	
Pivot word	**Candidate suffixes**
affect	ation, ations, ed, edly, eth, ting, ion, ionate, ionately, ions, s
affectation	s, ate, ately,
affected	ly
affection	ate, ately, s
affectionate	ly

C. Determine Lexicon Similarity

This score is measured with the help of a lexical similarity score, where morphologically related words show a high score for lexical similarity [37, 38]. The string similarity function maps a pair of words ps and qs to a real number r where the higher value of r denotes greater similarity between the word pair p and q after removing the suffix s. This metric presents the longest common prefix, which avoids any early mismatch while comparing the strings. We define similarity as $LongestCommonPrefix(LCP)$ between two words w_i and w_j taken from the corpus as shown in Eq. (4).

$$LCP = \frac{|L(w_{i,}w_{j,})|}{max(|w_i|, |w_j|)} \tag{4}$$

where $L(w_{i,}\, w_{j,})$ is the length of the common part of words w_i and w_j and denominator is the longer length of strings.

Algorithm 2 define the procedure to calculate the lexicon similarity and create the cluster of morphological related words.

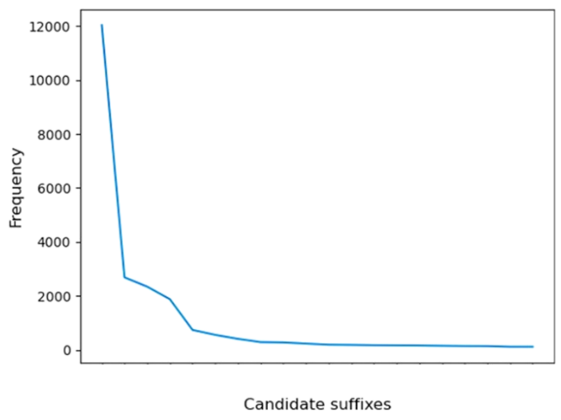

Figure 2. Example of candidate suffix with count/frequency (English corpus).

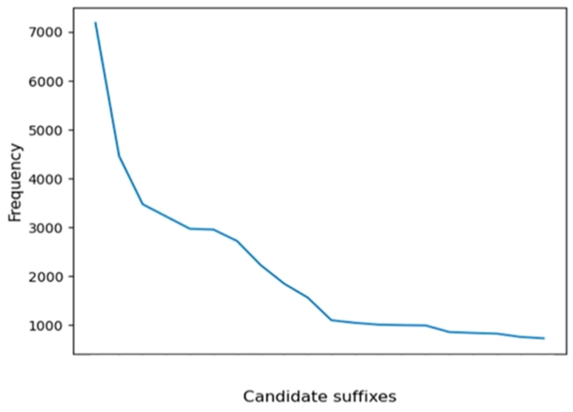

Figure 3. Example of candidate suffix with count/frequency (Urdu corpus).

D. Creating Morphological Classes

Morphological classes are constructed using suffix statistics and string similarity function. The list of input words is divided into several $C_1, C_2, C_3, \ldots, C_n$ based on a common prefix of a length 3. The process is described in Algorithm 2.

Algorithm 2: Grouping morphologically related words

```
1  Split the lexicon into initial classes $C_1, C_2, C_3, \ldots, C_n$
   such that each class has the first three letters in
   common
2  for each word in class $C_i = \{C_1, C_2, C_3, \ldots, C_n\}$ do
3  Let $P \leftarrow w_0$ is a pivot word
4    for each $w_j$ in $C_i$
5       Iteratively truncate the suffix from $w_j$ and
   compute
      LCP using equation Eq. (4)
6     If score < 0.8 then
         $w_j$ is included in $C_m$
7     else
         $P \leftarrow w_j$ and repeat steps 4 to 5.
8    end for
9  end for
```

E. Illustrative Example

The proposed stemmer identifies Morphological Clusters (MC) of similar words, then the smallest word is deduced as a stem. The stemmer correctly identifies different types of suffixes, induced rules of the English language (examples: “near”, “neared”, “nearer”, “nearest”, “nearing”, “nears” stem to near).

IV. Experimental Results

We have used three different evaluation metrics to measure the performance of the PWMStem stemmer with the existing stemmers. A brief description of these evaluation metrics and analysis of obtained result is given in the following subsection.

A. Suffix Removal Evaluation

We compare the stems produced by the stemmers with manually annotated words by human experts. The manual annotation process includes labeling each word with its true stem. The suffix removal results have been assessed in terms of accuracy (Eq. (5)), precision (Eq. (6)), recall (Eq. (7)), and F-score (Eq. (8)). These evaluation metrics are commonly used to assess the performance of stemmers [12, 16].

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \tag{5}$$

$$Precision = \frac{TP}{TP+FP} \tag{6}$$

$$Recall = \frac{TP}{TP+FN} \tag{7}$$

Here, TP is the number of correct stems produced by the stemmer; FP is the number of incorrect stems; FN is the number of words incorrectly un-stemmed by the stemmer; TN is the total number of correctly un-stemmed words returned by the stemmer.

$$F-score = \frac{2 \times precision \times recall}{precision+recall} \tag{8}$$

Using the URSTEM dataset, PWMStem outperforms other stemmers with the highest accuracy of 0.874 and an F-score of 0.908, as shown in Table V. Our stemmer achieves an accuracy of 0.876 and an F-score of 0.907 using the Humayoun dataset, which is better than the existing stemmers. The PWMStem method achieved an accuracy score of 0.877 and 0.874 for the F-score using the OANC dataset for the English language. With the English Lemma dataset, the values of accuracy and F-score were 0.78 and 0.837, respectively. Fig. 4 compares accuracy, precision, recall, and F-score for URSTEM. The results obtained from the Humayoun dataset are given in Fig. 5. Fig. 6 demonstrates the score for the English dataset OANC, and the results obtained using the English Lemma dataset are shown in Fig. 7.

TABLE V. Results of Selected Stemmers on Four Datasets

Method	Accuracy	Prec.	Recall	F-score
Urdu results for URSTEM dataset				
PWMStem	0.874	0.94	0.878	0.908
Alotaibi and Gupta [4]	0.715	0.856	0.721	0.783

Linguistica [27]	0.448	0.66	0.50	0.57
Jabbar *et al.* [12]	0.42	0.628	0.458	0.53
Akram *et al.* [28]	0.352	0.583	0.31	0.404
English results for OANC dataset				
PWMStem	0.877	0.922	0.831	0.874
Alotaibi and Gupta [4]	0.71	0.763	0.632	0.691
Linguistica [27]	0.55	0.55	0.71	0.62
Porter [14]	0.50	0.52	0.49	0.504
Lovins [15]	0.413	0.433	0.448	0.44
Urdu results for Humayoun dataset				
PWMStem	0.876	0.974	0.85	0.907
Alotaibi and Gupta [4]	0.606	0.846	0.547	0.665
Linguistica [27]	0.44	0.642	0.566	0.601
Jabbar *et al.* [12]	0.3528	0.576	0.368	0.449
Akram *et al.* [28]	0.49	0.605	0.50	0.54
English results for English Lemma dataset				
PWMStem	0.782	0.938	0.755	0.837
Alotaibi and Gupta [4]	0.57	0.89	0.48	0.62
Linguistica [27]	0.33	0.58	0.387	0.464
Porter [14]	0.44	0.69	0.44	0.54
Lovins [15]	0.56	0.80	0.53	0.64

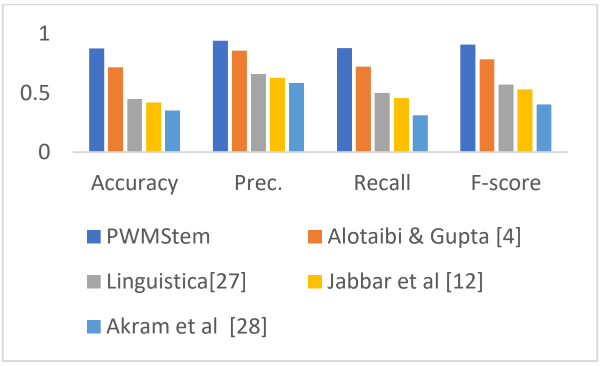

Figure 4. Results comparison for Urdu URSTEM dataset.

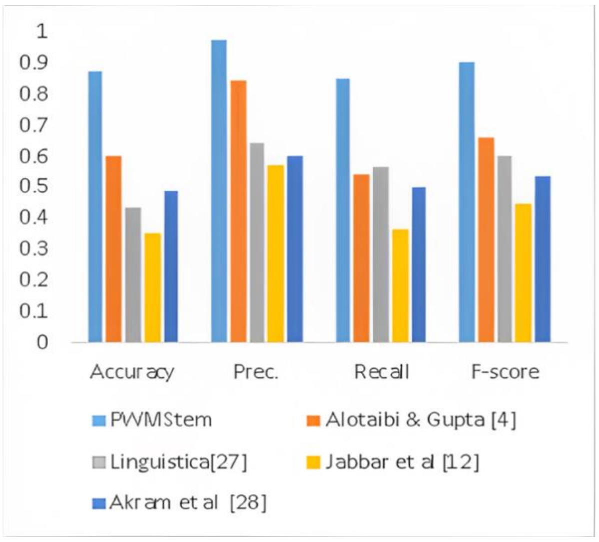

Figure 5. Results comparison for Urdu Humayoun dataset.

B. Stemming Errors

Paice [35, 36] presented the Over-stemming Index (OI), Under-stemming Index (UI), and their ratio, which is termed as Stemming Weight (SW) parameters, and the Error Rate Relative to Truncation (ERRT). Many researchers have used these metrics to assess the performance of their designed methods [41, 42]. To obtain the score of OI, Paice [35, 36] used Wrongly Merged Total (WMT) and Desired Non-merged Total (DNT) parameters. The WMT is used to measure the over-stemming errors, and it is computed via Eq. (9):

$$WMT_G = 0.5\sum_{i=1}^{fs} n_{si}(N_s - n_{si}) \tag{9}$$

N_s is the total number of stems in the stem group n_{si} is the number of stems obtained from thei^{th} concept group t is the number of groups that share the same stem. The stemmer may confuse some words of one group with another morphologically different group. Paice [35, 36] used the Desired Non-merge total (DNT), which is given by the following formula (Eq. (10)):

$$DNT_g = 0.5N_s(w - N_s) \tag{10}$$

where w is the total number of words. By summing WMT (Eq. (9)) and DNT (Eq. (10)) over all groups in the sample, we obtain the Global Unachieved Non-Merge Total $GWMT$ and Global Desired Non-Merge Total ($GDNT$), respectively. The OI is a ratio as given below (Eq. (11)).

$$OI = \frac{GWMT}{GDNT} \tag{11}$$

To calculate the error rate of UI, Paice [35, 36] introduced the Desired Merged Total (DMT) and Unachieved Merged Total (UMT) parameters. DMT represents the number of all pairs of words in the group and is given as follows (Eq. 12):

$$DMT_g = \frac{1}{2}N_s(N_s - 1) \tag{12}$$

where, N_s= the number of words in that group.

DMT =0, if a group contains only one word. (that's why the minimum number of items in a group must be two, as mentioned in sub section A, Corpus Development of Section III)

GUMT is defined by the following Eq. (13):

$$UMT_G = 0.5\sum_{i=1}^{fg} n_{gi}(N_g - n_{gi}) \tag{13}$$

where, fg is the number of distinct stems in the group g, and n_{gi} is the total number of cases of stem i in the group.

The sum of DMT (Eq. (12)) and UMT (Eq. (13)) overall groups give us the Global Unachieved Merge Total (GUMT) and Global Desired Merge Total (GDMT). Thus, these parameters defined the UI as follows (Eq. (14)):

$$\text{UI} = \frac{GUMT}{GDMT} \tag{14}$$

Stemming Weight (SW) refers to the ratio $\frac{OI}{UI}$. An aggressive stemmer strips too many affixes and thus has a higher value of OI than UI. On the other hand, a light stemmer removes a few affixes and hence has high UI and low OI. Error Rate Relative to Truncation (ERRT) refers to the values of (UI, OI) for a series of truncation lines from Trunc4 to Trunc7. The coordinates (UI, OI) for a stemmer should be below the truncation line, and such stemmers are known as "good" stemmers. ERRT is taken by stretching a line from the origin through the coordinates (UI, OI) point P until it intersects the truncation line at T (see Fig. 8). ERRT is computed by Eq. (15):

$$ERRT = \frac{length(OP)}{length(OT)} \tag{15}$$

The standalone value of OI and UI did not specify whether a stemmer is better or not because there is a tradeoff relationship between them. If suffix stripping rules are added or modified to reduce the under-stemming errors, these modifications will probably introduce additional over-stemming errors. Consequently, ERRT is a better metric that shows the performance of a stemmer. The lower score of ERRT showed a better stemmer. A better stemmer obtains ERRT closer to the origin, i.e., O, whereas the poor stemmer will be away from the origin.

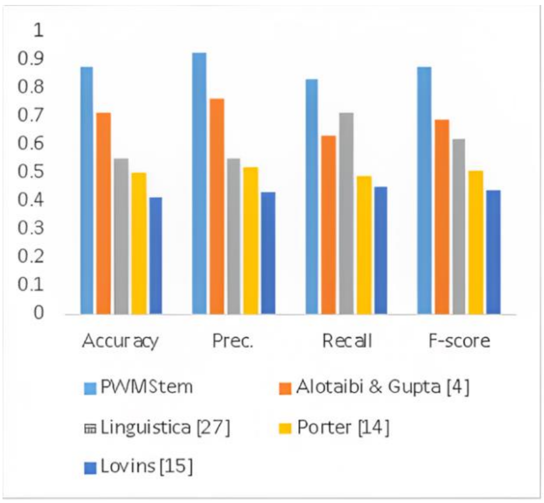

Figure 6. Results comparison for English OANC dataset.

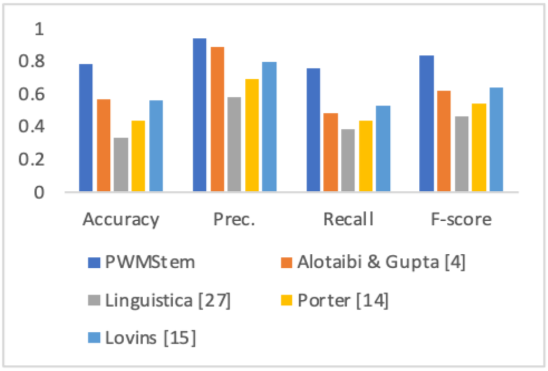

Figure 7. Results comparison for English Lemma dataset.

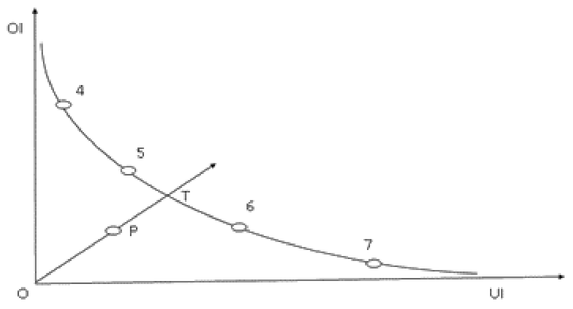

Figure 8. ERRT computation ([36] adapted).

To compute the relative accuracy of the stemmers, we use the ERRT line. It is useful for choosing the best stemmer in cases where one stemmer is better in under-stemming but worse in over-stemming. To calculate the ERRT, we created a baseline using truncation length by reducing the words in the word list to their n first letters where n is 4, 5, 6, and 7. The values of (UI, OI) any reasonable stemmers should be found between this line. The ERRT point of the 'best' stemmer is nearest the origin O as compared to the rest. The comparison results of Paice evaluation methods [35, 36] are mentioned in Appendix A. In this Section, we examine the error-based evaluation parameters [35, 36] of the PWM method and existing stemmers. The performance of the under-study stemmer is compared with baseline Trun4, Trun5, Trun6, and Trun7 (in Fig. 8).

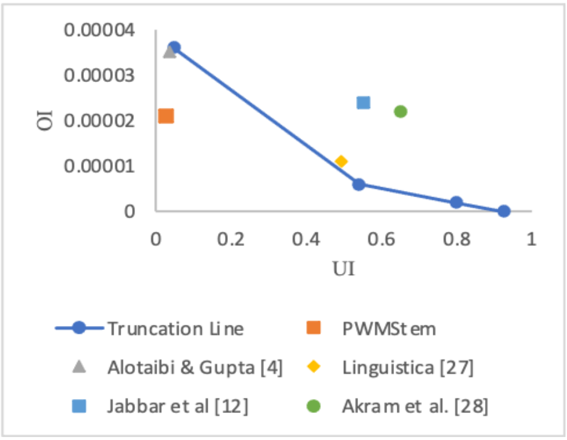

Figure 9. ERRT plot for Urdu language on URSTEM dataset.

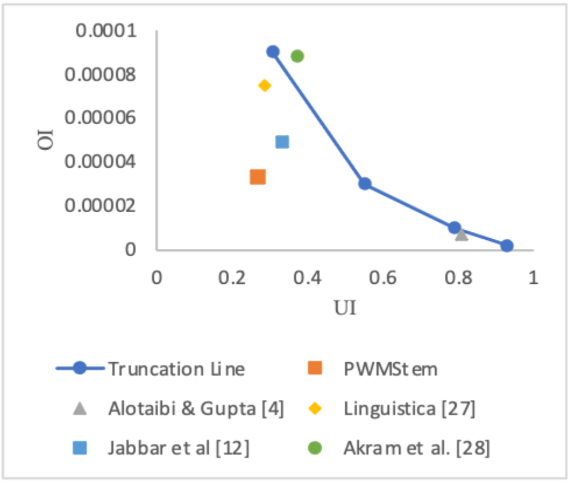

Figure 10. ERRT plot for Urdu language on Humayun dataset.

For the URSTEM dataset, the PWMStem produced a GUMT value of 4034, which is lower than existing stemmers. UI is 0.026479, OI is 0.000021, and ERRT is 0.61, which is the lowest score compared to the existing stemmers. The PWM stemmer also has the lowest score of UI with 0.267608 and 0.820 for ERRT. Alotaibi and Gupta [4] achieved the highest UI with 0.807471 using Humauoun dataset. Using the OANC dataset, PWMStem has the lowest value of GUMT with 1310 and GWMT with 4095 and has the lowest UI of 0.102089, OI 0.000015, and ERRT score is 0.498. For the English lemma dataset, the

PWM stemmer achieves the lowest GUMT with 5930 but the lowest GWMT for the porter [14] method, which is 6638.5. The PWM system has a lower value of UI and OI. In this dataset, our PWM stemmer is slightly better than the porter stemmer. Figs. 9–12 plots hypothetical UI and OI values of stemming operations taken from Urdu and English datasets. As shown in Fig. 9, only two stemmers are better than truncation stemmers, that is [4] and PWMStem. Among these two, PWMStem is closer to the origin O, as shown in Fig. 9. For the Humayoun dataset, three stemmers are below the truncation line, and our PWMStem is closer to the origin than other stemmers, as shown in Fig. 10. For the English language, UI and OI scores fall under the truncation line for all stemmers. This is given in Figs. 11 and 12. However, our PWMStem is closer to the origin O.

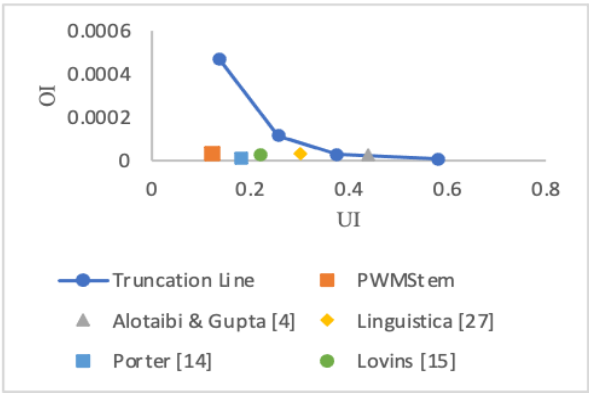

Figure 11. ERRT Plot for English language on English lemma dataset.

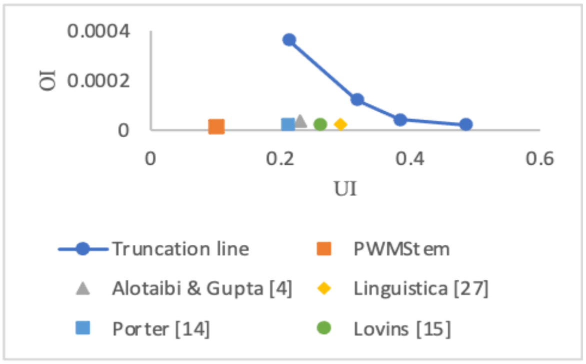

Figure 12. ERRT Plot for English language on OANC dataset.

C. Stemmer Strength Measurement

A stemmer strength indicates the average change in producing stem concerning a given word. We use the Mean Number of Words per Conflation Class (MWC) and Index Compression Factor (ICF) to measure the strength of the stemmer. MWC refers to the average number of words conflated to the common stem. For example, if the words "reached", "reaches", and "reaching" are stemmed to "reach", then this conflation class size is three. A score of one for MWC indicates the weakest stemmer, which shows no change in any letter from the stemmed word, and such a stemmer is called the weakest stemmer. A higher value for MWC indicates a stronger stemmer.

ICF is described as the ratio of the number of unique words before stemming, and the number of unique stemmed words after stemming. The ICF was calculated using Eq. (16). A high value of this metric denotes a stronger stemmer.

$$ICF = (\mathrm{n} - \mathrm{s})/n \qquad (16)$$

where,
n= The number of words in the corpus
s= The number of stems

For example, a corpus with 100,000 words (n) and 40.000 stems (s) would have an index compression factor of 60%. The third experiment is about the strength of the stemmer.

We use MWC and ICF metrics to measure the strength of the stemmer. Tables VI and VII depict the strength of the stemmers. Using these stemmer strength measures, it is possible to define the limits of stemmer strength. The strongest stemmer removes all possible affixes. The Maximum Strength (MS) is measured manually on the annotated data set, which is mentioned in Table VII. Proposed method achieved MWC of 3.7 and 3.0 using URSTEM and Humayoun datasets, respectively, as shown in Fig. 13. For the English OANC dataset, our method has the highest MWC of 3.46 and 2.0 for the English lemma dataset, as mentioned in Fig. 8. Fig. 14 portrays the highest score of ICF, with 73.0 on the Urdu dataset URSTEM and 66.0 for ICF using Humayoun dataset and OANC dataset. We achieved 71.1 ICF score, the best score in the experimental set. Using the English lemma dataset, MWC is 50 ICF, as presented in Fig. 10.

TABLE VI. THE MWC RESULTS COMPARISON FOR URDU AND ENGLISH

Datasets	PWMStem	Mujeeb *et al.* [4]	Lee *et al.* [27]	Jabbar *et al.* [12]	Rehman *et al.* [28]	Porter [14]	Lovins [15]	MS
URSTEM	**3.5**	3.5	2.7	2.1	1.7			3.5
Humayoun	**3**	2.7	3	1.9	2			3.4
OANC	**2**	1.9	2			2	1.9	2.1
English Lemma	**3.5**	2	3.4			3.1	3.2	3.8

TABLE VII. ICF RESULTS COMPARISON FOR URDU AND ENGLISH

Datasets	PWMStem	Mujeeb *et al.* [4]	Lee and Goldsmith [27]	Jabbar *et al.* [12]	Rehman *et al.* [28]	Porter [14]	Lovins [15]
URSTEM	**73**	73	63	53	43		
Humayoun	**68**	63	68	47	58		
OANC	**53**	50	53			53	49
English Lemma	**71**	50	70			68	69

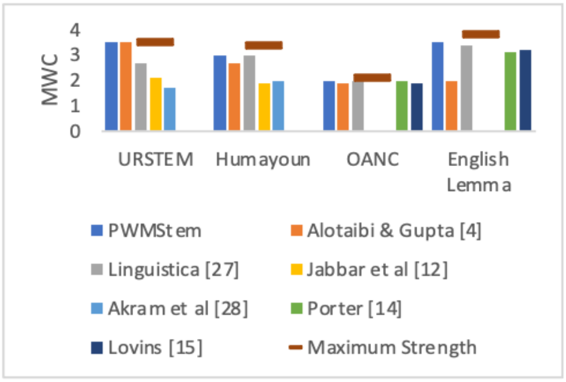

Figure 13. Comparison of the results of MWC for Urdu and English language.

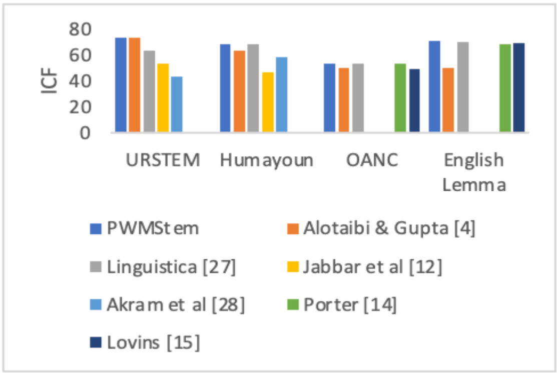

Figure 14. Comparison of the results of ICF for Urdu and English language.

V. Results and Discussions

Measuring the efficiency of a stemmer from different aspects such as accuracy, stemming errors, and strength of the stemmer is essential for several reasons [39, 40].

- Lemmatization groups the morphologically related words under a base dictionary form, a crucial step of NLP preprocessing in Machine Translation (MT), POS tagging, and Named Entity Recognition (NER). Accuracy measures the correctness of stem concerning manually annotated lemma. Lemmatizes also produce lemma. So, this assessment shows the ability of stemmer to replace the lemmatizes.
- The precision of an IR system is highly affected by under-stemming and over-stemming errors. IR system designers need a stemmer to address the vocabulary mismatch problem.
- The features are vital in NLP applications, including Text Classification (TC), Machine Translation (MT), and IR systems. The ICF and MWC metrics represent a feature reduction achieved through the stemming process.

The experimental results of the PWMStem show that it achieved better results as compared to the other linguistic [12, 14, 15, 28] and non-linguistic [4, 25–27] stemmers. We analyze the PWMStem from three aspects. Firstly, we determine how a system produces the correct stem using the manually annotated dataset. Linguistic stemmers are aggressive because these stemmers committed high under-stemming errors such as [ridiculousnesses] stems to [ridiculeness] by Akram *et al.* [28] and [ridiculous] produced by Jabbar *et al.* [12]. But the stem produced through the PWMStem method is [ridicule] which is the correct stem. That is why linguistic-based stemmers have low accuracy and F-score, as shown in Table IV. The performance of language-independent stemmers is usually lower because these stemmers produce the stem like [separation] to [invalid Urdu word] which is incorrect in most cases. On the other hand, as an example, our stemmer performed better and produced the correct stem [separately].

In the same way, the English linguistic stemmer [14] and language-independent stemmer Linguistica [25–27] stems (“voice”, “voiced”, “voices”, “voicing”) to “voice”, which reduces the accuracy and F-score of the stemmer. The method given in [4] produces two stems, “voice” and ‘voicing’ for the same group of words. In contrast, our method produces only the correct stem that is “voice”.

The second experiment is conducted to know how a stemmer committed stemming errors. For this, metrics proposed by Paice [35, 36, 39] are used. As demonstrated in Table VIII, the PWMStemmer committed the smallest number of under-stemming and over-stemming errors on the Urdu data set by achieving a UI score of 0.025693, OI of 0.000483, and ERRT of 0.61 for the Urdu dataset URSTEM. Our method also outperforms other stemmers on the English dataset with UI of 0.102089, OI of 0.000015, and ERRT of 0.498. The sample results of over and under-stemming for English in Table VIII.

TABLE VIII. Sample Result of Over-Stemming and under Stemming for English

References	Query Text	Error Types
	[‘photograph’, ‘photographed’, ‘photographing’, ‘photographs’] [‘photostate’, ‘photostated’, ‘photostating’, ‘photostats’, ‘photostatted’, ‘photostatting’]	
PWMStem	[‘photograph’, ‘photographed’, ‘photographing’, ‘photographs’] [‘photostat’, ‘photostated’, ‘photostating’, ‘photostats’, ‘photostatted’, ‘photostatting’]	No error
Alotaibi and Gupta [4]	[‘photograph’, ‘photographed’, ‘photographing’, ‘photographs’, ‘photostat’, ‘photostated’, ‘photostating’, ‘photostats’, ‘photostatted’, ‘photostatting’]	Over-stemming
Linguistica [27]	[‘photostat’, ‘photostat’, ‘photostats’, ‘photostated’] [‘photostatt’, ‘photostatting’]	Under-stemming
Lovins [15]	[‘photograph’, ‘photographed’, ‘photographing’, ‘photographs’] [‘photostat’, ‘photostats’, ‘photostatted’, ‘photostatting’] [‘photost’, ‘photostating’]	Under-stemming
Porter [14]	[‘photograph’, ‘photographed’, ‘photographing’, ‘photographs’] [‘photostat’, ‘photostats’, ‘photostatted’, ‘photostatting’] [‘photost’, ‘photostating’]	Under-stemming

For the last evaluation metric, we measure cluster-level conflation and corpus-level conflation. We utilize words' level conflation to measure MWC and ICF. Our method also outperforms other stemmers for Urdu and English regarding the MWC and ICF, as mentioned in Tables VII and VIII. We achieved a 3.7 MWC score for the Urdu language for the PWMStem and [4] methods. [28]'s stemmer is the weakest stemmer with the lowest MWC of 1.7. Concerning the ICF score, 73% is achieved on our stemmer and [4] on URDSTEM. For the Humayoun dataset, the weakest stemmer is the one by [12] with an MWC of 1.9 and ICF of 58.0, and the strongest stemmer is Linguistica. The PWMStem method produced an MWC of 3.0 and an ICF of 68.0.

The proposed stemmer is stronger on the OANCE dataset with MWC 3.47 and ICF 71.0, as shown in Tables VII and VIII. Linguistica [27] is the weakest stemmer that obtained the MWC 2.0 and ICF 50.0. For the English Lemma dataset, the weakest stemmer [4] with MWC of 1.9 (see Table VII) and ICF of 49.0 (given in Table VIII). On the other hand, our stemmer is stronger with MWC of 2.0 and 53.0 for ICF scores closer to the maximum strength of the stemmer. Linguistica and Porter [14] obtained 2.0 for MWC and 53.0 for ICF. The weakest possible affix removal stemmer would be one that changes no characters in any stemmed word. Such a stemmer would have one word per conflation class.

VI. Conclusion and Future Work Directions

Text mining is a challenging task that involves analyzing raw data. One of the easiest methods to analyze text mining is changing the textual files or documents into a standard dataset. This is a challenge for stemmers, and to the best of our knowledge, no standard stemmer addresses such issues. Hence, in this article, we designed an algorithm to overcome such issues.

This paper proposes a new language-independent method to conflate morphologically similar words to the common stem. LCP measure is used to determine the linguistic similarity to the pivot word after removing the suffix extracted based on the pivot word. Two datasets are used for each of the Urdu and English languages to perform a comparative evaluation of stemmers. Further, the results of our PWMStem stemmer are compared with two linguistic-based and two language-independent stemmers for each language. We computed the performance with three different evaluation methods, that is accuracy measurement, error estimate, conflation, and index compression factor. Finally, results suggest that the PWMStem algorithm is more effective than the other stemmers tested. The PWMStem method can decrease the size of terms vocabulary from 53% to 73% in TC and increase the performance of IR systems. The accuracy score shows that PWMStem may be used instead of a lemmatizer in ML, POS tagging, and Named Entity Recognition (NER) systems.

Although the PWMStem stemmer achieved acceptable performance results, there are some things that could be improved in its behavior and functionality. This study has concentrated only on lexical similarity at the word level. Further research would be required to address sentencing level or semantic similarity to improve the accuracy score by introducing semantic processing. Application of the PWMStem method on big datasets and its scope enhancement, including languages such as Arabic, German, French, and Turkish, is also part of future work. The big dataset usually includes many co-suffixes or words with a larger length of suffixes that need special treatment. Recent advances in AI and statistical methods urge using artificial neural networks and other machine learning methods to develop more efficient and high-performing stemmers. Recently, transformers and generative AI have shown huge progress in various applications. Applications of these algorithms can lead to high performance methods.

Appendix: Results Produced Using Paice's [35, 36] Evaluation Method

Method	**GUMT**	**GDMT**	**GWMT**	**GDNT**	**UI**	*OI*	*SW*	**ERRT**
				Urdu results for URSTEM dataset				
PWMStem	**4034**	152349	46190	2252629777	**0.026479**	0.000021	0.000774	**0.610**
Alotaibi and Gupta [4]	5458.5	152349	79017.5	2252629777	0.035829	0.000035	0.000979	1.03
Linguistica [27]	75141	152349	**24463.5**	2252629777	0.493216	**0.000011**	0.000022	0.993
Jabbar *et al.* [12]	84201	152349	54561	2252629777	0.552685	0.000024	0.000044	1.450
Akram *et al.* [28]	99061	152349	50174	2252629777	0.650224	0.000022	0.000034	1.532
Trun 4	7391	153823	81774.5	2272810353	0.048514	0.000036	0.000748	1
Trun 5	82424	153823	14192	2272810353	0.541021	0.000006	0.000012	1
Trun 6	121507.5	153823	3457.0	2272810353	0.797560	0.000002	0.000002	1
Trun 7	141009.5	153823	752.5	2272810353	0.925569	0.000000	0.000000	1
				Urdu results for Humayoun dataset				
PWMStem	**2880**	10762	1143	34488209	**0.267608**	0.000033	0.000124	**0.820**
Alotaibi and Gupta [4]	8690	10762	**237.5**	34488209	0.807471	**0.000007**	0.000009	1.057
Linguistica [27]	3097	10762	2584.5	34488209	0.287772	0.000075	0.000260	1.367
Jabbar *et al.* [12]	3598	10762	1703	34488209	0.334324	0.000049	0.000148	1.123
Akram *et al.* [28]	4003.5	10762	3030	34488209	0.372003	0.000088	0.000236	1.656
Trun 4	3312	10762	3101	34488209	0.307749	0.000090	0.000292	1
Trun 5	5937	10762	1037.5	34488209	0.551663	0.000030	0.000055	1
Trun 6	8503.5	10762	341	34488209	0.790141	0.000010	0.000013	1
Trun 7	10009.5	10762	77	34488209	0.930078	0.000002	0.000002z	1
				English results for OANC dataset				
PWMStem	**1310**	12832	**4095**	269327613	**0.102089**	**0.000015**	0.000149	**0.498**

Alotaibi and Gupta [4]	2968.5	12832	8929	269327613	0.231336	0.000033	0.000143	1.104
Linguistica [27]	3769.5	12832	6359	269327613	0.293758	0.000024	0.000080	1.045
Porter [14]	2731.5	12832	6177	269327613	0.212866	0.000023	0.000108	0.870
Lovins [15]	3356	12832	6595.5	269327613	0.261534	0.000024	0.000094	0.997
Trun 4	2734	12832	97829.5	269327613	0.213061	0.000363	0.001705	1
Trun 5	4085	12832	33058	269327613	0.318345	0.000123	0.000386	1
Trun 6	4935.5	12832	11315	269327613	0.384624	0.000042	0.000109	1
Trun 7	6226.5	12832	6058.5	269327613	0.485232	0.000022	0.000046	1
English results for English Lemma dataset								
PWMStem	**5930**	48052	19813.5	568049726	**0.123408**	0.000035	0.000283	**0.444**
Alotaibi and Gupta [4]	21129	48052	14510	568049726	0.439711	0.000026	0.000058	1.130
Linguistica [27]	14579	48052	17638.5	568049726	0.303400	0.000031	0.000102	0.842
Porter [14]	8703.5	48052	**6638.5**	568049726	0.181127	**0.000012**	0.000065	0.471
Lovins [15]	10656	48052	14775.5	568049726	0.221760	0.000026	0.000117	0.631
Trun 4	6608	48052	266227.5	568049726	0.137518	0.000469	0.003410	1
Trun 5	12404.5	48052	65325.0	568049726	0.258147	0.000115	0.000445	1
Trun 6	18071.5	48052	16304.5	568049726	0.376082	0.000029	0.000076	1
Trun 7	27995	48052	3751	568049726	0.582598	0.000007	0.000011	1

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Conceptualization: A.J., A.R.K., and M.I.; Methodology: T.S., S.A.; Software: M.I and N.E.; Validation: A.J. A.R.K., N.E., and T.S.; Writing—original draft preparation: A.J., S.I., A.R.K., and M.I.; Writing—review and editing: T.S., N.E.; Visualization: M.I. and T.S.; Supervision: A.R.K.; Project administration: T.S., S.I.; all authors have read and agreed to the published version of the manuscript.

Acknowledgement

This work was supported by Artificial Intelligence & Data Analytics Research Lab, CCIS Prince Sultan University Riyadh Saudi Arabia. The authors would like to thank Prince Sultan University Riyadh Saudi Arabia for the support of APC for this publication.

References

[1] S. Latif, F. Shafait, and R. Latif, "Analyzing LDA and NMF topic models for Urdu tweets via automatic labeling," *IEEE Access*, vol. 9, pp. 127531–127547, 2021.

[2] A. T. Azar, Z. I. Khan, S. U. Amin, *et al.*, "Hybrid global optimization algorithm for feature selection," *Comput. Mater. Contin.*, vol. 74, pp. 2021–2037, 2023.

[3] A. A. Laith, M. Shahbaz, H. F. Alaskar, *et al.*, "Arasencorpus: A semi-supervised approach for sentiment annotation of a large Arabic text corpus," *Applied Sciences*, vol. 11, no. 5, 2434, 2021.

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

[5] A. Rehman and T. Saba, "Performance analysis of character segmentation approach for cursive script recognition on benchmark database," *Digital Signal Processing*, vol. 21, no. 3, pp. 486–490, 2011.

[6] T. Saba, A. Rehman, A. Altameem, *et al.*, "Annotated comparisons of proposed preprocessing techniques for script recognition," *Neural Computing and Applications*, vol. 25, pp. 1337–1347, 2014.

[7] T. Saba, A. Rehman, and M. E. Boudihir, "Methods and strategies on off-line cursive touched characters segmentation: A directional review," *Artificial Intelligence Review*, vol. 42, pp. 1047–1066, 2014.

[8] H. Alshalabi, S. Tiun, N. Omar, E. A. Anaam, and Y. Saif, "BPR algorithm: New broken plural rules for an Arabic stemmer," *Egypt. Informatics J.*, vol. 23, no. 3, 2022.

[9] H. Alshalabi, S. Tiun, N. Omar, F. N. A. Aswadi, and K. A. Alezabi, "Arabic light-based stemmer using new rules," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 34, no. 9, 2021.

[10] 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, 2018.

[11] M. Harouni, M. S. M. Rahim, M. Al-Rodhaan, *et al.*, "Online Persian/Arabic script classification without contextual information," *The Imaging Science Journal*, vol. 62, no. 8, pp. 437–448, 2014.

[12] 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, 2018.

[13] S. Khan, W. Anwar, U. Bajwa, and X. Wang, "Template based affix stemmer for a morphologically rich language," *Int. Arab J. Inf. Technol.*, vol. 12, no. 2, pp. 146–154, 2015.

[14] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980.

[15] J. B. Lovins, "Development of a stemming algorithm," *Mech. Transl. Comput. Linguist.*, vol. 11, pp. 22–31, 1968.

[16] J. Singh and V. Gupta, "A novel unsupervised corpus-based stemming technique using lexicon and corpus statistics," *Knowledge-Based Syst.*, vol. 180, pp. 147–162, 2019.

[17] M. Kasthuri, S. B. R. Kumar, and S. Khaddaj, "PLIS: Proposed language independent stemmer for information retrieval systems using dynamic programming," in *Proc. the 2nd World Congr. Comput. Commun. Technol.*, 2017, pp. 132–135.

[18] C. Chavula and H. Suleman, "Morphological cluster induction of Bantu words using a weighted similarity measure," in *Proc. SAICSIT'17: The South African Institute of Computer Scientists and Information Technologists*, 2017, pp. 1–9.

[19] J. Singh and V. Gupta, "An efficient corpus-based stemmer," *Cognit. Comput.*, vol. 9, no. 5, pp. 671–688, 2017.

[20] T. Brychcín and M. Konopík, "HPS: High precision stemmer," *Inf. Process. Manag.*, vol. 51, no. 1, pp. 68–91, 2015.

[21] 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, 2013.

[22] 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, 2013.

[23] J. H. Paik, D. Pal, and S. K. Parui, "A novel corpus-based stemming algorithm using co-occurrence statistics," in *Proc. the 34th Int. ACM SIGIR Conf. Res. Dev. Inf. Retr.*, 2011, pp. 863–872.

[24] 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, 2007.

[25] J. Goldsmith, "Unsupervised learning of the morphology of a natural language," *Computational Linguistics*, vol. 27, no. 2, 2001.

[26] J. Goldsmith, "An algorithm for the unsupervised learning of morphology," *Nat. Lang. Eng.*, vol. 12, no. 4, pp. 353–371, 2006.

[27] J. L. Lee and J. A. Goldsmith, "Linguistica 5: Unsupervised learning of linguistic structure," in *Proc. 2016 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol*., 2016, pp. 22–26.

[28] Q. U. A. Akram, A. Naseer, and S. Hussain, "Assas-Band, an affix-exception-list based Urdu stemmer," in *Proc the 7th Workshop on Asian Language Resources*, 2009, pp. 40–47.

[29] M. Humayoun, R. M. A. Nawab, M. Uzair, S. Aslam, and O. Farzand, "Urdu summary corpus," in *Proc. the 10th International Conference on Language Resources and Evaluation*, 2016, pp. 796–800.

[30] T. Saba, A. Rehman, and G. Sulong, "Cursive script segmentation with neural confidence," *Int. J. Innov. Comput. Inf. Control*, vol. 7, no. 7, pp. 1–10, 2011.

[31] T. Saba and F. A. Alqahtani, "Semantic analysis-based forms information retrieval and classification," *3D Research*, vol. 4, no. 3, pp. 1–6, 2013.

[32] Z. Hussain, S. Iqbal, T. Saba, *et al.*, "Design and development of dictionary-based stemmer for the Urdu language," *Journal of Theoretical & Applied Information Technology*, vol. 95, no. 15, 2017.

[33] S. L. M. Sainte, B. S. Alnamlah, N. F. Alkassim, and S. Y. Alshathry, "A new framework for Arabic recitation using speech recognition and the Jaro Winkler algorithm," *Kuwait J. Sci*., vol. 49, 2022.

[34] S. Hussain. Finite-state morphological analyzer for Urdu. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.128.1603&rep=rep1&type=pdf

[35] C. D. Paice, "Method for evaluation of stemming algorithms based on error counting," *J. Am. Soc. Inf. Sci*., vol. 47, no. 8, pp. 632–649, 1996.

[36] C. D. Paice, "An evaluation method for stemming algorithms," in *Proc. the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1994, pp. 42–50.

[37] K. Neamah, D. Mohamad, T. Saba, and A. Rehman, "Discriminative features mining for offline handwritten signature verification," *3D Research*, vol. 5, pp. 1–6, 2014.

[38] T. A. Khan, "Morphological integration of Urdu loan words in Pakistani English," *English Lang. Teach*., vol. 13, no. 5, 49, 2020.

[39] 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, 2017.

[40] T. Saba, A. Rehman, and G. Sulong, "Improved statistical features for cursive character recognition," *International Journal of Innovative Computing, Information and Control*, vol. 7, pp. 5211–5224, 2011.

[41] 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, 2017.

[42] F. N. Flores and V. P. Moreira, "Assessing the impact of stemming accuracy on information retrieval—A multilingual perspective," *Inf. Process. Manag.*, vol. 52, no. 5, pp. 840–854, 2016.