High performance stemming algorithm to handle multi-level inflections in Urdu Language

Abdul Jabbar
COMSATS University Islamabad (CUI)
Sajid Iqbal (Sajid.iqbal@bzu.edu.pk)
Bahauddin Zakariya University
Manzoor Ilahi
COMSATS University Islamabad (CUI)

Research Article

Keywords: Stemming, Lemmatizer, Natural Language Processing, Urdu Language, Text Mining, Information Retrieval, Stemmer Evaluation

Posted Date: May 12th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1529588/v2

Abstract

Stemming is an essential step in varied Natural Language Processing (NLP) applications. It is used to reduce different variants of the query words to a standard form to avoid the vocabulary mismatch issue in Information Retrieval (IR) systems. There are a lot of stemming algorithms in English language, but Urdu NLP still in its infancy. A stemmer develops and evaluates in a language cannot be efficiently used for other languages . Urdu has the highly inflected and complex morphological structure, and most of the current Urdu stemmers remove a few numbers of affixes and leads to inefficient results. In this context, according to the best of our knowledge, this is a first comprehensive effort towards the proposition and evaluation of a novel Urdu Text Stemmer (UTS) that cope with multi-level inflection and derivation forms in Urdu text. The experimental evaluation of the proposed scheme has been conducted on the text-based and words-based custom developed corpus. The proposed stemming technique is rigorously evaluated and compared with state-of-the-art stemming algorithms. Experimental results demonstrate that UTS outperforms existing Urdu stemmers and achieves an accuracy of 94.92% and 91.8% on word and text corpus, respectively.

1 Introduction

Text stemming is a complicated and crucial step in many query systems, indexing, web search engines and IR systems (El Mahdaouy, Gaussier, and El Alaoui 2019; Bechikh Ali, Haddad, and Slimani 2019; Y. Chen et al. 2019; L.-C. Chen 2018; Djenouri et al. 2021) document classification (Alhaj et al. 2019) ,(Razali et al. 2020; Tubishat et al. 2019) and linguistic feature extraction (L.-C. Chen 2018). It provides the benefits of reducing the storage requirements by truncating redundant terms (Labani et al. 2018). In fact, it increases the matching possibility for comparing documents and unifying the vocabulary process (Nguyen, Duong, and Cambria 2019; Kumar, Bhanodai, and Pamula 2019). In fact, the stemming is a computational process that reduces all conflated words to the same root or stem by stripping derivational and inflectional affixes(Jabbar et al. 2016). For example, the English words like ‘consisted’, ‘consistency’, ‘consistent’, ‘consistently’, ‘consisting’, and ‘consists’ can be reduced to ‘consist’.

In general, there are three types of stemming algorithms such as affix stripping (linguistic knowledge based) (Akram, Naseer, and Hussain 2009), corpus base (Paik and Parui 2011), and a Multi-step Urdu (MU) stemmer (Jabbar et al. 2018). In Affix stripping methods, a set of rules is developed based on the morphological structure of the language. Some affixes are truncated from one or both sides of the root/stem to extract a stem. The Corpus-based stemmer consists of the table lookup method. In the table lookup method, each set of morphological- related words and their corresponding stems is stored in structured form. However, this process consumes a lot of space to maintain the manual updates for each word. Contrarily statistical methods are used to extract various features of the words in statistical stemming. Examples of some adopted statistical methodologies are n-gram(Husain, Ahamad, and Khalid 2013),(Brychcín and Konopík 2015) and Hidden Markov Models (Momenipour, Of, and undefined 2016, n.d.). Whereas, (Bimba et al. 2015) presented a hybrid stemmer, in which affix striping and table lookup methods were used to obtain the stem.

Urdu is a semantic language with a composite morphological structure. It is different from most of the western languages (Jabbar et al. 2016). The existing stemmers for the Urdu language commit over-stemming, under-stemming, and miss-stemming errors (Jabbar et al. 2016). Consequently, these errors decrease the efficiency of stemming algorithms. Whereas over-stemming occurs when two words of different classes are stemmed to the same root, for instance, the Urdu word [hathon/hands] and [hathi/elephant] are mistakenly merged such as [hath/hand]. Furthermore, the under-stemming takes place when two words of the same group must not be stemmed to the same root, for instance, when the stemmer fails to conflate the words [Morton/mortals] and the word [amwaat /motels] to the common root form as [mout/mortal]. Mis-stemming is defined as taking off the affix that is an actual part of the word, for example, stemming the Urdu word [bukhaar/fever] to [khaar/barb]. We notice that the mis-stemming error is frequently encountered in the Urdu language. The commonly used Urdu prefix [bay] and suffix [ye] may often be the actual letter of the word, for example, the Urdu words [baqya/the restand]and [larki/girl].

In this paper, an Urdu Text Stemmer (UTS) is proposed to clip the multi-level inflected, derivation, as well as Mohmil words (meaningless words). The recommend algorithm consists of compound word reduction, truncating the prefixes, suffixes, co-suffixes, infixes, and Mohmil words phases. The existing Urdu stemmers fail to capture the stem from multilevel inflection, derivation, and Mohmil words. For instance, bigram words having co-suffix such as [soobay daar/officer of a province], a Mohmal word as a suffix like [chori chaakari/stolen]. According to our knowledge, it is the first comprehensive effort to handle multi-levels inflections and derivations in the Urdu language. Two types of data sets (word corpus and text corpus) are used to rigorously evaluate the efficiency of the UTS. The uniqueness of this paper among exiting works is summarized as below:

- Multi-level inflections are handled by UTS, while current Urdu stemmers do not consider it, for instance, [baikhlaq/Well-mannered] possesses prefix [ba] and some infixes letters, after striping, the affixes and derived stem is [khulq/politness].
- Co-suffixes are not handled in any present Urdu stemmer; however, UTS copes with them, for example, [rishte dar/relatives] is stemmed as [rishta/relation].
- Existing Urdu stemmers do not remove the Mohmil words; however, UTS deletes the Mohmil words and extract the stem from meaningful words, such as [chori chkari/thieft] is stemmed to [chor/thief].

This paper is organized as follows: Section II provides a brief background of the Urdu language grammar and morphology. Section III describes the existing work carried out in the same direction. Whereas section IV elaborates on the proposed technique and evaluation datasets. The experimental results and the discussion are presented in section V. Finally, section VI concludes the paper.

2 Background

Urdu is known as one of the major languages of the world (Mehmood et al. 2019). It has been ranked second among 2,301 major languages of the world after the English with 527 million speakers around the globe (Qureshi et al., n.d.). The rapid increase in the quantity of Urdu web documents over the recent years has created a dire need for improving the performance of IR and text classification systems. Therefore, developing an accurate stemmer is crucial for the automatic processing of Urdu language (Jabbar et al. 2018).

In comparison to the English language, Urdu uses a non-concatenated way to derive the morphemes which are interwoven to form words. In the concatenated languages, the position of affixes and stems are coupled by concatenating morphemes. Urdu morphemes are interwoven in such a way that it is hard to obtain the stem from pattern less Urdu words. For example, the [Motain /deaths], [Mouton/deaths] [maiyat /dead body] and [amwaat /deaths] are derived from the root word [mout /death]. Hence it is challenging to extract the stem from this linear decomposition principle by state-of- the-art algorithms.

Urdu is a highly inflected language that is written from right to left in contrast to the English language. New words are coined by derivation and compounding (Jabbar, Iqbal, and Khan 2016). In the derivation, affixes (prefixes and/or suffixes) are attached to the root word to coin a new word. Both prefixes and/or suffixes are concatenated to the root to modify the meaning. In the Urdu, prefixes are added to the right of the stem, and suffixes are added to the left, such as (suffix) [chohti ye] + (root) [ittafaq/ unity] +(prefix) [na] and become [na-itefaqi/ Disunity].

TABLE 1. Inflection examples in Urdu language.

Examples	Transform	Method
[waalid/father] [walidain/parents]	Dual to singular	suffixation
[jail/prison] [jail khanah jaat/ prisons]	Noun transform	Multilevel suffixation
[arki/girl] [larkian/girls]	Singular to plural transform	suffixation
[marz/a disease] [amraz/diseases]	Singular to plural (Broken/ irregular) transform	infixation

TABLE 2. **Examples of loan prefixes in Urdu.**

Affixes	Prefixes	Words
Persian prefixes	[paish]	[paishkhaima/ precursor]
Hindi prefix	[un]	[unparh/illiterate]
Arabic prefix	[la]	[lazawaal/Everlasting]
English prefixes	[city]	[city naazim/city coordinator]

In compounding, two completely independent and meaningful words and affixes are joined together to make a compound word [18] [20]. For instance, [khush akhlaq/ well-mannered] in which both words are meaningful, but in another compound word like [ibadat gaah/ house of worship], standalone the (affix) [gaah] has no meaning. However, if such meaningless word is attached with some meaningful word, it can produce new meanings. The meanings of [ibadat/ worship] are changed when affixes are attached. Hybrid compound word is another form of compound words (Deutsch, Velan, and Michaly 2018) in which two words of different languages are added to make compound word such as (English word) [tax]+(Urdu word) [ghunda / hooligan] become a compound word [ghunda tax/ hooligan tax]. Reduplication (Qureshi et al., n.d.) (Haq 1996) is also a form of the compound word in which both words are slightly different to each other such as [roti woti/ bread] where [woti] is a Mohmil word. In the Urdu language, grammatical changes occur through suffixation as well as infixation(Schmidt, n.d.; Baloch, n.d.). The examples of suffixation and infixation in Urdu are given in table 1. Multilevel inflection and derivations can also cause a change in Parts of Speech (PoS) group.

TABLE 3. Examples of loan suffixes in Urdu.

Affixes	suffixes	Words
Persian suffixes	[daani]	pot][chayedaani/Tea
Hindi suffixes	[hat]	[chiknahat/oily]
Arabic suffixes	[yat]	[shakhsiyat/personality]
English suffixes	[store]	store/grocery store][karyana

Urdu nouns are modified to signify possession, plurality, and agency like English nouns. However, Urdu verbs are modified more expansively than English verbs. Around 60 different forms can be generated (Rizvi and Hussain 2005). Urdu is highly Persianised and Arabicized because Arabic and Persian have significantly influenced in terms of vocabulary and sentence structure (Labani et al. 2018; Parveen 2014). Urdu has a few native affixes, most are borrowed from Persian and Arabic (Islam 2012). The examples of borrowed affixes are given in table 1 and table 2. According to table 1 and table 2, it is difficult for existing approaches to recognize the tough stems. To address this issue for Urdu language, that is truncation of Arabic and Persian affixes, we combined the template-based approach, affix striping, and reference lookup.

3 Related Work

Several stemming algorithms have been developed in many languages (Jabbar et al. 2018; Brychcín and Konopík 2015; Porter 1980; Savoy 1993; Bacchin, Ferro, and Melucci 2002; Taghi-Zadeh et al. 2015). However, most stemmers are developed for a few major languages like English, Arabic and Chinese. A very effective affix stripping approach based on automatic purging of affixes from root word has not been considered in the development of stemmers for Urdu language (Jabbar et al. 2019). In the subsequent text, some state-of-the-art stemmers are described and analyzed.

Lovins (Lovins 1968) proposed a context-sensitive, longest- match stemming algorithm with the exception list for the English language. It was among popular stemmers introduced in early studies. It works in two steps: the first step is a basic stemming procedure in which suffix is removed using predefined sixty suffix rules and the second step is recoding procedure in which stems are converted into valid root word. Porter (Porter 1980) presented a root-based stemmer without exception list. Moreover, it does not deal with prefixes. It iteratively removes the suffixes until the termination condition is met. It comprises of five steps; the first step removes the plural and past participant suffixes. In step 2 to 4, suffixes are truncated using predefined suffix removal rules. In step 5, recoding is performed, and the stem/root is derived. Porter stemmer (Porter 1980)and its variants have been employed in various natural language processing task (Chintala, In, and undefined 2013, n.d.; Patil et al., n.d.). A non-iteration based stemming method is proposed by Dawson (Bulletin and undefined 1974, n.d.). as an extension of Lovins (Lovins 1968) work. It uses a large list of 1200 suffixes comprehensively and arranges the suffixes according to longest length first and so on. It is very fast due to non-iterative nature. However, it is complex due to its comprehensive list of suffixes (Savoy 1993; Anjali and Jivani, n.d.) suffix stripping algorithm consists of two steps. The first step is the inflectional analysis in which four different tables of suffixes corresponding to the four grammatical categories (noun, adjective, verb, and adverb) are created. The derivational suffixes are removed according to the grammatical category In the second step. Bimba et al. (Bimba et al. 2015) cohabiting the affix striping and table lookup approach to develop a stemmer for the Hausa language. Koirala and Aman Shakya (Koirala and Shakya 2020)built a rule-based stemmer for Nepali language. Suryani (Suryani et al. 2018) design a rule based stemmer for Sundanese language, which iteratively removes the affixes.

Alshalabi (Alshalabi et al. 2022) developed rules to extract the root words from Arabic word. Alnaied (Alnaied, Elbendak, and Bulbul 2020) recognized the affix by predefined eight pattern and obtain the root. Alshalabi (Alshalabi et al. 2021) proposed Arabic stemmer that removed the prefix and suffix with respect to the words' length instead of pattern. Atwan et al. (Atwan 2019) presented Arabic light stemmer that outperform in Arabic retrieval system. (Bessou and Touahria 2019) developed ESAIR (Enhanced Stemmer for Arabic Information Retrieval) using linguistic resources such as Arabic words dictionary. Al-Kabi et al. (Al-Kabi et al. 2015) proposed an Arabic stemmer that uses defined prefix and suffix lists to identify the true prefix or suffix which are further used to recognize the infixes. Finally, recoding step is performed to derive the correct stem. Abuata and Al-Omari (Abuata and Al-Omari 2015) classified the Arabic language into three variant: Classical Arabic, Modern Standard Arabic (MSA) and Colloquial or Dialectal Arabic. In this study, the authors presented a stemmer for the Gulf dialect Arabic to extract the stem after removing all the affixes that appeared in MSA such as [Alef- Lam], [waw-noon], vowels ([alif], [Yaa], [waw]). For instance, the Arabic word [tasho foon/You will see] is stemmed to [shof/look] by removing the prefix [taa] and suffix [waw/noon]. Whereas Rahimi (Rahimi 2015) used lookup table and affix removal techniques to develop Persian stemmer. Table lookup consists of Mokassar words and their respective stems, and affixes (prefix/suffix) list to recognize and delete the affixes (prefix/suffix). Aslamzai and Saad (Aslamzai and Saad 2015) developed a Pashto stemmer using a rule-based approach. Meitei (Meitei, Purkayastha, and Devi 2015) presented a Manipuri stemmer using affix striping and stem word dictionary. Kaur and Preetpal Kaur Buttar (Kaur and Buttar 2019) combined the table lookup and suffix deletion approach to extract the stem of Punjabi verb. Shah et al. (SHAH et al. 2016) designed a stemmer for the Sindhi language that used a lexicon and linguistic rules to obtain the stem of the query word. Aldabbas et al. (Aldabbas et al. 2016) presented an Arabic stemmer by combing the prefix and suffix removal and using patterns matching approaches. Afterward, the regular expressions are constructed based on derived patterns. Then, these regular expressions are used to extract the stem from the query word. Both stems obtained by affix removal and regular expression are checked in the Microsoft word dictionary. If both have the same meaning, then the potential stem is extracted. El-Defrawy (El-Defrawy, El-Sonbaty, and Belal 2016) presented a context-sensitive Arabic rule-based stemmer. That works in two steps: in the first step, query word is matched to possible pattern and potential stem is produced. In the second step, the root frequency

map is constructed to derive the possible stem by minimizing the frequency map of roots. Zeroual (Zeroual et al. 2017) used the pattern matching techniques to derive the stem of the Arabic word. For instance, the word [kataba/to write], [kaatib/writer], [makotuwb/written], makotab /office], [makotabaq/library] share the same stem [ktb/ books]. Azman (Azman 2019) combines bottom-up approach and top of bottom to retrieve the root of the Arabic words from Tree-hierarchized structure of Arabic. The head node contains the root and child depict the morphological form of the root verb. Yusuf and Wahid, Yusuf et al (Yusuf, Yunus, and Wahid 2020) and query expansion techniques to improve the performance of Arabic IR system. Mustafa and Rashid (Mustafa and Rashid 2017) proposed an improved rule-based stemmer for the Kurdish language. The stemmer tokenizes the query text, after that normalization is performed in which an Arabic letter such as [yaa] is replaced with another Arabic letter [yeh]. Using a precede list of prefixes and suffixes, Kurdish words are removed. Finally, the stop words are removed, and the stem words list is obtained. Whereas, Rouibia (Rouibia, Belhadj, and Cheragui 2017) integrated a pattern matching technique to remove affix and dictionary-based approaches for developing stemmer. First, the query text is tokenized and further normalized, then pattern matching, affix striping, and dictionary base techniques are applied respectively. Abainia et al (Abainia, Ouamour, and Sayoud 2016) proposed a rule-based stemmer for the Arabic language. They developed it in six steps and in each step specific affixes (prefixes and/or suffixes) are removed to derive the stem. Saeed et al (Saeed et al. 2018) proposed iterative rule based stemmer that removes the longest affixes (suffix and prefixes) from query words. Bölücü and Can (Bölücü and Can 2019) proposed context sensitive stemmer combined with POS for Agglutinative Languages.

The existing Urdu text stemmers focus on the challenge of processing the Urdu language morphology. Akram et al. (Akram, Naseer, and Hussain 2009) developed Assas-Band Urdu stemmer and defined the affix (prefix and/or suffix) and affix exception lists to remove the affixes and extract the stem from them. Khan et al (S. A. Khan et al. 2012) presented an Urdu stemmer without affix exception lists and utilized a predefined affix list to remove the affixes. Vishal Goyal and Lehal (Kansal, Goyal, and Lehal 2012) developed an Urdu stemmer that produced the list of possible stems using appropriate affix rules. They presented a database of the stems with their frequency. The possible stem is searched in the database and high-frequency stems are derived. Another study by Gupta and Mathur (Gupta, Joshi, and Mathur 2015) presented the Urdu stemmer without exception list. They used the list of predefined affixes, and if true affix is identified then the stem is derived after truncating the affix. Husain et al. (Husain, Ahamad, and Khalid 2013) used n-gram approach to generate the suffixes that are chopped off using frequency-based and suffix-based length. Ali et al(M Ali, Khalid, and Saleemi 2014) defined the affix and stop words list. If a query word is not a stop word, then an appropriate affix is found to truncate the affixes. Khan et al. (S. Khan et al. 2015a) presented an Urdu stemmer using a template-based approach. It defined the template to identify infixes. If template matches with query word, then the corresponding rules are applied to extract the stem. Gupta and Mathur (Gupta, Joshi, and Mathur 2013) proposed the stemmer that initially checks the query word in exception and stop word lists. If the query word is not found in both lists, then true affixes are removed to derive the stem. Ali et al. (Mubashir Ali et al. 2016) defined the infixes handling rules that were ignored in their prior work (M Ali, Khalid, and Saleemi 2014) Hussain et al. (Z. Hussain et al., n.d.) proposed dictionary based stemmer for the Urdu language. Ali et al. (Mubashir Ali, Khalid, and Aslam 2017) used the patterns to recognize and remove the infixes. They used them in Urdu short text classification. Jabbar et al. (Jabbar et al. 2018) proposed a MU stemmer that extracted the bigram compound words from the text and derived the stem.

In the light of the above, the motivation in the present article is to devise a linguistic knowledge-based stemmer that handle the multilevel inflections and derivations in Urdu language.

4 Methodology

The proposed Urdu Text Stemmer (UTS) is presented in this section. The UTS is based on removing suffixes, co-suffixes, prefixes, infixes and Mohmil words from the query words to extract the stem as shown in figure 1. UTS is based on seven main steps that are described in the following paragraphs.

First three steps are preprocessing, reducing compound words, and transforming the token (obtained in step 2) into unigram. Then, steps 4 and 5 manipulate the unigram words and derive the stem by clipping the affixes if any.

Handling trigram words, multi-level affixes including Mohmil words is a hard task. First, UTS extracts the content words from the query text. Then, it removes the affixes to obtain a stem. For query word, compound word reduction, suffix removal, and recoding, prefix and suffixes removal, infixes matching and removal, and table lookup approaches are integrated and applied in a sequence to extract the stem. Examples of applying the proposed algorithm on words corpus and on text corpus are presented in figure 2 and figure 3, respectively. The steps listed in algorithm 1 provide the overview of proposed methodology.

A. STEP 1: PREPROCESSING AND TOKENIZATION

To create a vocabulary set, query text is tokenized. Tokenization marker list is mentioned in Appendix D. The tokenization is based on hard space and stop words ([ka/of], [par/on/at], [se/to]).

B. STEP 2: COMPOUND WORD REDUCTION

The second step consists of removing the compound affixes and Mohmil words. In the Urdu language, stem able compound words (bi-gram and trigrams) are formed by adding the affixes (prefix and/or suffix) or coined by adding Mohmil words as an affix with the root word. For example:

- The Urdu trigram words [ghairtar- biyatYafta/ untrained] in which (Suffix) [Yafta] + (root) [tarbiyat]+ (prefix) [ghair], and derived stem is [train]
- The Urdu trigram compound word [jail khanah jaat/ the prisons] that consist of (Suffix) [jaat] + (suffix) [khanah] + (root) [jail], and obtained stem is [jail/ the prison]

- The Urdu bigram word [baikhlaq/ Well- mannered] possesses prefix [ba] and some infixes letters. After striping these affixes, the derived stem is [khulq/politness].
- The Urdu bigram word [aqalmand/ wise] contains (Suffix) [mand] + (root ([aqal/ wis- dom] and extracted stem is [aqal/wisdom]
- An Urdu compound word always contain prefix]hama-waqt]. (Root) [waqt/ time] + (prefix) [hama], and the produced stem by the system is [waqt/time].
- The Urdu compound word [ghalat salat/ wrong] in which (Mohmil words) [salat] + (root) [ghalat/ wrong], and [ghalat/ wrong] is extracted as a stem.

The processed query word in this step may not be a final stem, for instance [mardana waar/ by male], and produced word is [mardana/ male] that is not a final stem as the word [mardana/ male] still has [ana] suffix, and to remove this, the extracted word (i.e., [mardana/ male]) is passed to the next step. If compound word reduction rules do not match, then unchanged query word passes to the next step. Examples of rules Mohmil word reduction is given in table 4, and the list is given in Appendix B

TABLE 4. Example of Mohmil affixes identification and removal

Input Word	Criteria	Stem
[pani-vani/ water]	Second-word start with [wao] and the rest of the letters are alike in both words.	[pani/ water]
[akeladkela/ alone]	Second-word start with [dal] and the rest of the letters in both words are identical.	[akela/ alone]

C. STEP 3: SPLIT TOKEN INTO UNIGRAM

In this step, the token received by the previous step is split based on hard space, and unigram words list is constructed. For example, if tokens obtained from prior step (step 1) are [salah baba noor] then it is splitted based on hard space into two words [baba noor /name of a person] and [salah/ annual].

D. *STEP 4: SUFFIX REMOVAL AND RECODING*

The procedure takes a query word from the preceding step and checks the existence of the suffix based on word's length. If the suffix is found, then the conditions related to that rule are applied and the stem is derived. Otherwise, original word is passed to the next step. When the number of rules attached to the criteria is more than one, then the derived stem is verified from the SWL (Stem Words List), for instance, a word ending in has three rules:

- Rule 1: [waday/promises] [wad] by removing the suffix [bri ye]
- Rule 2: [waday/promises] [wada] by replacing suffix with [alif]
- Rule 3: [waday/promises]

[wadah/promise] by replacing suffix with [he]

In the above case, the system produces three possible stems using the above rules and each is checked in SWL, if found, then is added to the EL (End List) as stem. Otherwise, the original word is passed to step 5. In the above example three possible stems are produced, in which only [wadah/promise] one is found in SWL, which is added to the EL as a stem. The exceptions, of these rules are handled by table lookup approaches such as [karaye/rental] is stemmed to [kiraya/rent]. The complete list of suffix removal and recoding rules can be found in our research work (Jabbar, Iqbal, and Khan 2016) .

Abbreviations used in algorithm:

- SWFTL: Stop words free text list
- CWR: Compound words reduction rules
- OWL: one words list
- FSL: Final Stem List
- SWL: Stem word list
- SL: Suffix List
- PL: Prefix list

ALGORITHM 1: stem_produce Function

read query text

String [] function stem_produce(string query_text)

// step 1

Preprocessing – *Tokenize query_text where non-Urdu or stop words are removed and remaining words are added in SWFTL*

// Step 2

For each *token in SWFTL*

If *token is bi-gram or trigram*

apply CWR and update SWFTL at that index

Else *Keep token unchanged, and go to step 3*

End for

// Step 3

Tokenize *SWFTL by hard space & add produced tokens to the OWL, and go to step 4*

// Step 4

For each *word in OWL*

If *suffix removal and recoding rule is match*

apply the rule and add to FSL

Else if *given word remains unchanged and go to step 5.*

// Step 5

Else If *the affix removal rule is applicable to the word*

apply the rule and add the stem to FSL. go to step 6

// step 6

Else If *the infix removal rule is applicable*

apply the rule and add the stem to FSL. go to step 7.

// step 7

Else If *the word found in reference lookup table*

retrieve the corresponding stem and add to FSL

Else *add the original word to the FSL*

End for

E. STEP 5: CIRCUMFIXES, PREFIX OR SUFFIX REMOVAL

The procedure initially checks the true circumfixes (both prefix and suffix) by predefined Prefix List (PL) and Suffix List (SL). If true circumfixes are found, they are removed to get stem. For example: [nakhushgawaar/Unpleasant] stems to [khush/Happy]. If true circumfixes are not identified, then true prefixes are checked, if found, then prefix is truncated, and the stem is added to the FSL. For instance, [nojawan/younger] is stemmed to [jawan/young]. If a true prefix is not found, then check for true suffix, if found, then remove the suffix and add a FSL; otherwise, original word is passed to the next step. For example: [zamindar/landlord] is stemmed to [zamin/ land].

To avoid the under stemming and over stemming, affixes are removed according to the length of query words. Minimum query words length is set to four characters and minimum produced stem length will be three characters and if two-character stem is derived, it is verified from the SWL. Here, we deal with maximum 8-character long affixes. These suffixes are arranged in descending order and removed with the longest matching first. The example of suffixes is given in table 5 and prefixes are shown in table 6. A complete list of these affixes can be found in Appendix D and Appendix E.

F. STEP 6: INFIXES HANDLING

This procedure identifies the infix letters by using pre- defined patterns (see table 7). If infixes are found then their corresponding rules are applied, and stem is added to an FSL.

On the contrary, if no infixes are identified, then original word is passed to the next step. When several rules are attached to a pattern then the obtained stem is verified from the SWL, for instance, the pattern matched with Urdu word [abdaan/ bodies] produces two stems:

- Rule 1. [abdaan/ bodies] [bdan/body], remove first and fourth letter [alif]
- Rule 2. [abdaan/ bodies] [bdah], remove first and fourth letter [alif] and substitute [he] at the end of the word.

The exception of these rules is handled by references lookup table, for instance, [ahsas/sensitives] where corresponding stem is [ehs/a sense of] and [adaad/numbers], where [adad/number] is the stem.

Table 5. Example of Urdu suffixes.

Suffixes	List
	List 1
	List 2
	List 3
	List 4
	List 5
	List 6
,	
	List 7
,	List 8

Table 6. Examples of Urdu prefixes.

Prefixes	List
	List 1
	List 2
	List 3
	List 4
	List 5
	List 6
	List 7
	List 8

Table 7. Example of infix handling rules.

Pattern	No. of Stemming Rules	Rules	Examples
If the Word length is five and first and fourth letters are [alif]	2	Remove first and fourth letter [alif]	[ehkaam /orders] stem to [hukum /order]
		Remove first and the fourth letter [alif] and sub- stitute [hey] at the end of the word.	[lttehaf /gifts]
		first,	[tohfa /gift]
If the Word length is six, and first and fifth letters are [alif] and third letter is [te]	1	Remove second and fifth letters from the query word.	[ekhtataam/ ends] stem to [khatam /end]

G. *STEP 7: REFERENCES LOOKUP*

This step takes a query word and checks the existence of the query word in the table lookup, if found, then corresponding stem is returned. Otherwise, query word is included in the FSL. For instance, [asaatzaa/teachers] has its appropriate stem [ustaad/ teacher].

5 Evaluation Criteria

Sseveral evaluation criteria have been suggested In literature to evaluate the strength and accuracy of stemming algorithms (Sirsat, Chavan, and Mahalle 2013; Frakes and Fox 2003; Paice 1994). To measure the correctness and strengthen of UTS, Sirsat et al. (Sirsat, Chavan, and Mahalle 2013) evaluation metrics, as well as the precision, recall and F-measure metrics are used.

The chosen evaluation measures (Jabbar et al. 2018; S. Khan et al. 2015b) are described as follows:

Accuracy = (TP + TN) / (TP + TN + FP + FN)

Recall = (TP) / (TP + FN)

Precision= (TP) / (TP + FP)

F1(recall, precision) = 2(Precision*Recall) / (Precision + Recall)*

Sirsat et al. (Sirsat, Chavan, and Mahalle 2013) criterion is very compelling for assessing the strength and accuracy of a stemming algorithm. The following parameters are used to evaluate the strength and accuracy of the stemmer (Sirsat, Chavan, and Mahalle 2013).

Index compression factor (ICF)

ICF indicate the percentage by which a collection of distinct words is reduced by stemming. The ICF is defined as

ICF = n - s

Where,

n = Total number of words before stemming

s = Number of words after stemming

Word Stemmed Factor (WSF)

It is an average number of words that a stemmer has stemmed. The threshold value is 50

WSF = WS / TW * 100

Where,

WS indicates the number of stem words,

TW stands for the total number of words

Sample correctly stemmed word factor (CSWF)

The value of CSWF shows the accuracy of the stemmer. The minimum threshold is 50

CSWF = (CSW / SW) * 100

Where,

CSW = correctly stem words, whereas

WS refers to the total number of stemmed words

Average Words Conflation Factor (AWCF)

AWCF is the average number of variant words of different conflation class that is stemmed correctly to the stem/root. To calculate AWCF, we first compute the number of unique words after conflation, which is calculated as follows

NWC = S - CW

Where,

S shows the number of distinct stems after stemming,

CW refers the number of incorrectly stemmed words.

Then, AWCF is computed as:

AWCF=(CSW-NWC)/(CSW)*100

To evaluate the performance of the proposed stemming algorithm, a series of experiments is conducted.

5.1 Corpus description

In order to evaluate UTS, we constructed the dataset which contains text fragments, including news articles (politics, literature, science, and technology) collected from BBC Urdu (W1, n.d.) and (W2, n.d.) containing 20000 words, including stop words, verbs, adverb, adjectives, nouns, proper nouns, punctuation marks, English words, numbers and special symbols. Words corpus consists of 56074 Urdu words containing uni-gram, bi- gram, tri-gram compound words, broken plural words and words with infixes. The dataset titled USED (Urdu Stemmer Evaluation Dataset) is collected form the following sources. (It will be online when the paper will be accepted):

- Four Urdu grammar books (Haq 1996; Board 2010; Baloch, n.d.; UEP 2014).
- Resources provided by (S. Hussain 2004) on Urdu morphology,
- Online resources: Urdu online encyclopedia (W3, n.d.), CLE Urdu words list (W4, n.d., 4)
- CLE Urdu high frequency words list (W5, n.d.).

The words corpus have been preprocessed, by the following steps:

- Removal of stop words, punctuations, numbers, and symbols.
- Deletion of English and French characters.
- Elimination of Urdu diacritics.

The text documents are related to 4 topics and each topic is represented by 5 texts with different lengths (Table 8), which lead to a total 20000 words. The text corpus feed the stemmer without preprocessing and token the text using token marker mentioning appendix.

Table 8
USED dataset description

Text corpus	Topic	No of articles	Total Words
1	Politics	5	6500
2	Literature	5	4600
3	Science	5	5300
4	Technology	5	3600
Total		**20**	**20000**

5.2 Results and discussion

To measure the performance of proposed stemmer we involved the human experts to annotate the words with the corresponding actual stem. They are native Urdu speakers with a relevant qualification. Obtained annotations are cross validated by each other and results are used for the rule extraction which leads to the development of stemmer. The stemmer is applied to the raw data. The results produced are compared with human expert annotations.

In this subsection, we compare the obtained results of UTS with existing Urdu stemmers Assas-Band stemmer (Akram, Naseer, and Hussain 2009) and MU stemmer (Jabbar et al. 2018). The selection of stemmers is based on multiple factors. Assas-Band stemmer (Akram, Naseer, and Hussain 2009) stemmer has high accuracy among the rule-based Urdu stemmer so selected as representative of Urdu rule-based stemmers. Similarly, MU stemmer (Jabbar et al. 2018) is better among infixes removal stemmers (Mubashir Ali, Khalid, and Aslam 2017; S. Khan et al. 2015b). Likewise(Akram, Naseer, and Hussain 2009) is a statistical stemmer that is tested on Urdu text, and Husain et al. (Z. Hussain et al., n.d.) is sole stem word dictionary based Urdu stemmer.

Table 9
UTS confusion matrix description

Characteristics	Words corpus	Text corpus
TP	51788	10347
FN	2391	674
FP	457	268
TN	1438	198

Confusion matrix statistics obtained after applying stemmer on both corpora is shown in Table 9.

Finally, Table 10 and Table 11 show the performance measures of UTS compared with state-of-the-art Urdu stemmers. Majority of the existing Urdu stemmers (e.g. (Akram, Naseer, and Hussain 2009; Husain, Ahamad, and Khalid 2013; S. Khan et al. 2015b) are evaluated on the word corpus. Usal stemmer (Gupta, Joshi, and Mathur 2015) also works on textual data. But it uses hard space to identify the words boundary. In Usal stemmer, compound words are wrongly split into two unigram words, therefore, compound word remains unstemmed. For instance, [yeh ibadat gaah hai/this is a place of worship] that is tokenized form of the compound word [ibadat gaah/ place of worship] into two unigram word [ibadat] is a root word, [gaah] is a suffix, consequently compound word [ibadat gaah/ place of worship] remains unstemmed.

The effectiveness of UTS is measured by using words corpus and text corpus as mentioned in Table 10. The major attributes to measure the performance is prediction of correctness of query word's stem. We conducted two different experiments and performed comparisons in term of accuracy on word corpus. The experimental results are presented in Fig. 5 and Fig. 6. Husain et al. (Z. Hussain et al., n.d.) claimed a higher accuracy of 94.85% on 3600 Urdu words, and stem words dictionary size is not mentioned. Assas Band stemmer (Akram, Naseer, and Hussain 2009) tested their stemmer on corpus consisting of 21757 Urdu words and achieved 92.97% accuracy. Their stemmer removed prefixes and /or suffixes but did not handle the infixes. Husain et al. (Husain, Ahamad, and Khalid 2013) proposed N-gram stemmer to truncate suffixes and ignored the prefixes and infixes. This stemmer obtained 84.27% purity on test size of 1200 Urdu words that are extracted from the E-mail corpus. Khan et al. (S. Khan et al. 2015b) used template to handle the infixes, but no mechanism is described to handle the exception of defined rules and template less Urdu words. For instance, a pattern is defined by Khan et al (S. Khan et al. 2015b), if an Urdu word has four characters and the third letter is [vao], the third letter is removed to extract the stem, but this rule is violated in case of [juloos/ procession] and [husool/ acquisition] to obtain the stem. This stemmer is evaluated on corpus consisting of 19351 words and claimed precision and recall are 89.95% and 96.08 respectively. The MU stemmer (Jabbar et al. 2018) assessed on both words and textual data and obtained a recall value of 99% and 95.586% precision.

Table 10
Performance comparison of UTS with state-of-the-art stemmers as reported in relevant papers

Stemmer	Corpus	No. of words	Used method	In %age			
				Acc.	Recall	Prec.	F1 score
Text Stemmer (UTS)	Words corpus	56074	Hybrid	94.92	99	95.58	97.32
	Text corpus	20000	-	91.8	97.48	93.88	95.65
MU stemmer (Jabbar et al. 2018)	Words corpus	56074	Multi-step	92.97	99	93.57	96.26
	Text corpus	20000	-	90.33	97.43	92.35	94.82
Hussain et al. (Hussain et al.2017)	Words corpus	3600	Dictionary based	94.85	-	-	-
Assas- Band stemmer (Akram, Naseer, and Hussain 2009)	Words corpus	21757	rule-based	91.2	-	-	-
Husain et al. (Husain, Ahamad, and Khalid 2013)	Words corpus	1200	Statistics (N-gram)	84.27	-	-	-

Table 11
Performance comparison using standard dataset

Sirsats' evaluation Metrics (Sirsat and Mahalle 2013)	Assas Band stemmer (Akram and Hussain 2009)	MU stemmer (Jabbar et al. 2018)	Urdu Text Stemmer (UTS)
Total words (TW)	56074	56074	56074
No. of distinct words after stemming (S)	26597	25233	24970
Index Compression Factor (ICF)	53	55	55.47
No. of stemmed words	55128	54012	54179
Words Stem factor (WSF)	98.31	96.32	96.62
correctly stem words (CSW)	49238	50693	51788
correctly stem words Factor (CSWF)	89.32	93.86	95.59
No. of distinct words after conflation (NWC)	24079	23795	23532
Average conflation words factor (AWCF)	51.10	53.06	54.56

MU stemmer (Jabbar et al. 2018) extracted the bigram word from textual data to produce stem, for example, the Urdu sentence [yeh ibadat gaah hai/this is place of worship], after eliminating the [yeh/this] and [hai/is], the compound word [ibadat gaah/ place of worship] is extracted and the suffix [gaah] is removed to obtain the stem [ibadat]. However, their stemmer fails to deal with Mohmil words, and multilevel inflection and derivation. Whereas UTS achieved recall and precision of 99% and 93.95%, respectively. These results are shown in Fig. 4. On text data set, the MU stemmer (Jabbar et al. 2018) yields 90.33% accuracy, followed by UTS that achieved accuracy of 91.8% as mentioned in Fig. 4. The results obtained on the same data set, show that UTS achieved the better performance (see Fig. 4 to 6) than existing state of the art MU stemmer (Jabbar et al. 2018) and Assas-Band stemmer (Akram, Naseer, and Hussain 2009) specifically, existing Urdu stemmers have caused some under-stemming errors for certain groups of words that hold multi-level inflection and derivation, for examples: Bigram words having co-suffix [thaanaydaar/the officer of a police station], corresponds to the mistaken stem [police stations].

The Urdu bigram word [taleem Yafta/ educated], possessed suffix [yafta] and some infixes letters, existing stemmer commit under stemming errors and produced [taleem/education] as a stem. Bigram words having a Mohmil word as an affix [samjhabujha/understand] cannot be stemmed. Trigram words having prefixes and suffix along with infixes, such as [gher taleem yafta/uneducated], possess prefix, suffix, and infix. The existing Urdu stemmers produce incorrect stem, i.e., [taleem/education]. Trigram words having co-suffix [jail khanah jaat/ the prisons] and produce the incorrect stem [jail khanah/ the prison].

Assas-Band stemmer (Akram, Naseer, and Hussain 2009) stemmer cannot handle the infixe cases that is why its performance is lower as mentioned in Table 11 and Table 12 and Fig. 5. Whereas UTS extracts the correct stem and it's obtained CSWF is significantly higher 95.59% from MU stemmer (Jabbar et al. 2018) obtained score 93.86% and Assas-Band stemmer (Akram, Naseer, and Hussain 2009) show lowest CSWF score 89.32% as reflected in Table 11 and Fig. 5.

In the second experiment, we use text corpus and compare the achieved accuracy with (Jabbar et al. 2018). Again, UTS outperform than MU stemmers (Jabbar et al. 2018) (see Fig. 6). The reason is, that MU stemmer (Jabbar et al. 2018) does not deduct the affixes from the token of three words size such as the obtained token [sarmaya kaari maliyat/ worth of investment] consists of a compound word [sarmaya kaari/investment], in which [kaari] is an affix and [sarmaya/capital] is a stem. The compared stemmers with the proposed one do not remove the Mohmil affixes and multi-level affixes as shown in Table 12 and the incorrect produced stem in the column is underlined. In the Table 12, we can notice that all the words having Mohmil suffix [chaakari], [cheet] are not stemmed. Whereas, in case of multi-level affix [mardana waar/manly] under stemming errors are committed by MU stemmer (Jabbar et al. 2018) and Assas-Band stemmer (Akram, Naseer, and Hussain 2009) UTS is the first Urdu stemmer that handles the multi-level inflections and Mohmil words reduction, as shown in Table 12. UTS has obtained 95.5 % CSWF while MU stemmer (Jabbar et al. 2018) achieved 93.8 % and Assas-Band stemmer (Akram, Naseer, and Hussain 2009)obtained lowest CSWF score 89.32%. Assas-Band stemmer (Akram, Naseer, and Hussain 2009) blindly removed the affixes and achieved highest WSF [98.31%] score than their competitor. The ICF achieved by UTS is 55.47% which is higher than the competitor with 55% value and 51%. as shown in table 13 and Fig. 6.

The performance of proposed UTS is comparatively higher with respect to CSWF and AWCF score as shown in Table 11. Therefore, from the obtained results using Sirsats (Sirsat, Chavan, and Mahalle 2013) evaluation method, we show that the UTS provides better results in terms of performance, strength, and accuracy.

Table 12
Example of produced stem by state of art Urdu stemmers

Query Words	Actual stem	Assas- Band stemmer (Akram and Hussain 2009)	Multi-step stemmer (Jabbar et al. 2018)	Urdu Text Stemmer (UTS)

The time complexity of the proposed algorithm is $\varvec{O}(\varvec{n})$. Because the execution time is directly proportional to the size of the input. Step 4 to 7 is executed in the nested loop which is based on the number of rules defined for the step. The fixed number of rules adds a constant factor in time complexity. Moreover, lower order terms and constants are ignored (Cormen et al. 2009).

Similarly, space complexity also remains linear. At the start of the algorithm, data is loaded, which is then reduced in coming steps. Rule lists used to stem the words are of a fixed size, which adds a constant space complexity that can be ignored in space complexity analysis (Cormen et al. 2009). The time and space complexity of the UTS is better than MU stemmer (Jabbar et al. 2018) which exhibits $O(n^2)$. Other exiting Urdu stemmer's time and space complexity is not provided, so the comparison is not possible.

Although the better efficiency to produce stem has been achieved by UTS; however, there are some deficiencies such as it may mistakenly stem (False Positive) the proper noun, for instance, [Irshad/Name of a person] is wrongly stemmed to [rushad/ guidance]. The reason is that there is no mechanism in the Urdu language to identify the proper nouns. The Mohmil compound words that are not split by hard space such as [khana wana/ the meal], patternless words and confusing words that may use as a root word or as an affix, cause the wrong stemming cases (False Negative). Although we handle such cases by table lookup approach and stem words list, it will depend upon the table lookup and stem words list entries. In case of text corpus, the accuracy is low as compared to words corpus (see Fig. 4) due to the improper tokenization and identification of proper noun, for instance, the Urdu sentence [Mohammad Haris Hussain taleem Yafta larka hai /Mohammad Haris Hus- sain is an educated boy] in which compound word [taleem Yafta/ educated] is not properly extracted due to the proper noun [Mohammad Haris Hussain] and remains un-stemmed or wrong stem is produced. The reason is that in such case hard space is used as a delimiter, as a result, compound words such as [taleem Yafta/ educated] became two independent words [taleem/education] and suffix [yafta]. In this situation, the suffix that becomes independent word is not removed but [taleem/education] is stemmed to [ilam/knowledge].

6 Conclusion

In this paper, we have proposed a novel multi-level in- flection and derivation handling stemmer (named UTS) for the Urdu language. According to best of our knowledge it is the first stemmer that considers the multi-level inflections in the Urdu language. The evaluation of the UTS shows that considering the multi-level inflection in Urdu stemmers improve the accuracy and performance of the stemming process. As a result, the UTS outperforms the state-of-the-art Urdu stemmers. Given this, the UTS has achieved an accuracy of 94.92% on word corpus and 91.8% on text fragments corpus. Moreover, we achieved ICF of 55.47%, WSF of 96.62%, CSWF of 95.59%, and AWCF is 54.56. Finally, this research's findings may help develop NLP tools in the domain of text mining, IR, text summarization, document indexing, spelling checker, parser, thesaurus, and dictionary. For future works, we plan to investigate the Urdu word morphology more deeply, particularly the word having infixes. Moreover, the segmentation process can be enhanced by adding more context-sensitive rules to further improve stemming performance.

Declarations

We, all authors, declare that the manuscript is original, has not been published elsewhere and it has not been submitted simultaneously for publication elsewhere. We wish to confirm that there are no known conflicts of interest associated with this publication. We confirm that the manuscript has been read and approved by all named authors and that there are no other persons who satisfied the criteria for authorship but are not listed. We further confirm that the order of authors listed in the manuscript has been approved by all of us.

Ethical Approval and Consent to participate

Conflict of Interest

The authors declare that they have no conflict of interest.

Ethical Approval

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

Consent for publication

We, all authors, give consent for publication of this article.

Availability of supporting data

The datasets generated and/or analyzed during the current study are available here: https://github.com/abduljabbar2017/Urdu-stemmer-dataset

Competing interests

The authors declare that they have no competing interests

Funding

This work is not funded by any agency.

Authors' contributions

Mr. Abdul Jabbar design and implement the algorithm, develop the dataset, and performed the experiments as well as write the manuscript. Dr. Sajid Iqbal and Dr. Manzoor Ilahi supervise this work and provided critical feedback and helped shape the research, analysis, and manuscript.

Acknowledgments

I would like to express my heartfelt thanks to the Masood Jhandir Research Library, Sardar pur Jhandir Mailsi, District Vehari, Punjab, Pakistan (jhandir.com) for providing us relevant, and important resources required for this research

Authors' information

MR. ABDUL JABBAR, is a Ph.D. scholar at the Department of Computer Science, COMSATS University Islamabad, Main campus, Pakistan. Previously, he did his master's in computer science from Institute of Southern Punjab, Multan. He has published more than 4 articles in international journals. His research interests include machine learning, text mining, artificial intelligence, and Urdu natural language processing.

DR. SAJID IQBAL, has done his Ph.D from University of Engineering and Technology, Lahore, Pakistan. He is an Assistant Professor at Department of Computer Science, Bahauddin Zakariya University, Multan, Pakistan. He has authored more than 20 research papers in local and international research journals. His research interests include Computer Vision, Natural Language Processing and Medical Image Analysis.

DR. MANZOOR ILAHI, did his master's in computer science in the year 1998 from Gomal University, D.I Khan. Later, in the year 2005, he was awarded CIIT Scholarship for Ph.D Studies at GSCAS, Beijing , P.R China. After completion of his Ph.D in Computer Science, he rejoined CIIT in the year 2009 as Assistant Professor in the Department of Computer Science. Currently, he is working as an Associate Professor in the same Department. His major areas of interest are Mining Data Streams, Ubiquitous Data Mining, Intrusion Detection, Distributed Data Mining, Machine learning and data mining, Real-Time system, and Text mining.

References

1. Abainia, Kheireddine, Siham Ouamour, and Halim Sayoud. 2016. "A Novel Robust Arabic Light Stemmer." Journal of Experimental & Theoretical Artificial Intelligence 29 (3): 557–73. https://doi.org/10.1080/0952813x.2016.1212100.
2. Abuata, Belal, and Asma Al-Omari. 2015. "A Rule-Based Stemmer for Arabic Gulf Dialect." Journal of King Saud University - Computer and Information Sciences 27 (2): 104–12. https://doi.org/10.1016/j.jksuci.2014.04.003.
3. Akram, Qurat-ul-Ain, Asma Naseer, and Sarmad Hussain. 2009. "Assas-Band, an Affix-Exception-List Based Urdu Stemmer." *Proceedings of the 7th Workshop on Asian Language Resources - ALR7*. Association for Computational Linguistics. https://doi.org/10.3115/1690299.1690305.
4. Aldabbas, Omar, Ghassan Kanaan, Motasim Albdarnah, Riyadh Alshalabi, Mohammed A Shehab, and Nizar Mahyoub. 2016. "Technique of Regular Expression for Arabic Light Stemmer." International Journal of Advanced Studies in Computers, Science and Engineering 5 (11): 175.
5. Alhaj, Yousif A., Jianwen Xiang, Dongdong Zhao, Mohammed A.A. Al-Qaness, Mohamed Abd Elaziz, and Abdelghani Dahou. 2019. "A Study of the Effects of Stemming Strategies on Arabic Document Classification." IEEE Access 7: 32664–71. https://doi.org/10.1109/ACCESS.2019.2903331.
6. Ali, M, S Khalid, and MH Saleemi. 2014. "A Novel Stemming Approach for Urdu Language."
7. Ali, Mubashir, Shehzad Khalid, and Muhammad Haseeb Aslam. 2017. "Pattern Based Comprehensive Urdu Stemmer and Short Text Classification." IEEE Access 6: 7374–89.
8. Ali, Mubashir, Shehzad Khalid, M Haneef, Waheed Iqbal, Armughan Ali, and Ghayur Naqvi. 2016. "A Rule Based Stemming Method for Multilingual Urdu Text." International Journal of Computer Applications 134 (8): 10–18. https://doi.org/10.5120/ijca2016907784.
9. Al-Kabi, Mohammed N, Saif A Kazakzeh, Belal M Abu Ata, Saif A Al-Rababah, and Izzat M Alsmadi. 2015. "A Novel Root Based Arabic Stemmer." Journal of King Saud University - Computer and Information Sciences 27 (2): 94–103. https://doi.org/10.1016/j.jksuci.2014.04.001.
10. Alnaied, Ali, Mosa Elbendak, and Abdullah Bulbul. 2020. "An Intelligent Use of Stemmer and Morphology Analysis for Arabic Information Retrieval." Egyptian Informatics Journal 21 (4): 209–17.

11. Alshalabi, Hamood, Sabrina Tiun, Nazlia Omar, Fatima N AL-Aswadi, and Kamal Ali Alezabi. 2021. “Arabic Light-Based Stemmer Using New Rules.” *Journal of King Saud University-Computer and Information Sciences*.
12. Alshalabi, Hamood, Sabrina Tiun, Nazlia Omar, Elham abdulwahab Anaam, and Yazid Saif. 2022. “BPR Algorithm: New Broken Plural Rules for an Arabic Stemmer.” Egyptian Informatics Journal.
13. Anjali, Ms, and Ganesh Jivani. n.d. “A Comparative Study of Stemming Algorithms.” *Kenbenoit.Net*. .
14. Aslamzai, Sebghatullah, and Saidah Saad. 2015. “Pashto Language Stemming Algorithm.” Asia-Pacific Journal of Information Technology and Multimedia 04 (01): 25–37. https://doi.org/10.17576/apjitm-2015-0401-03.
15. Atwan, Jaffar. 2019. “Arabic Text Light Stemmer Arabic Information Retrieval View Project.” *Researchgate.Net*. http://www.meacse.org/ijcar.
16. Azman, Bakeel. 2019. “Root Identification Tool for Arabic Verbs.” IEEE Access 7. https://ieeexplore.ieee.org/abstract/document/8686325/.
17. Bacchin, Michela, Nicola Ferro, and Massimo Melucci. 2002. “The Effiectiveness of a Graph-Based Algorithm for Stemming.” *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-36227-4_12.
18. Baloch, Dr. Sohail Ahmad. n.d. <background-color:#FFD9B3;idirection:rtl;> </background-color:#FFD9B3;idirection:rtl;>. First. : .
19. Bechikh Ali, Chedi, Hatem Haddad, and Yahya Slimani. 2019. “Empirical Evaluation of Compounds Indexing for Turkish Texts.” Computer Speech & Language 56: 95–106. https://doi.org/10.1016/j.csl.2019.01.004.
20. Bessou, Sadik, and Mohamed Touahria. 2019. “AN ACCURACY-ENHANCED STEMMING ALGORITHM FOR ARABIC INFORMATION RETRIEVAL.” *Arxiv.Org*, November. https://doi.org/10.14311/NNW.2014.24.007.
21. Bimba, Andrew, Norisma Idris, Norazlina Khamis, and Nurul Fazmidar Mohd Noor. 2015. “Stemming Hausa Text: Using Affix-Stripping Rules and Reference Look-Up.” Language Resources and Evaluation 50 (3): 687–703. https://doi.org/10.1007/s10579-015-9311-x.
22. Board, P. T. 2010. <background-color:#FFD9B3;idirection:rtl;> </background-color:#FFD9B3;idirection:rtl;>. Lahore: Punjab Textbook Board.
23. Bölücü, Necva, and Burcu Can. 2019. “Unsupervised Joint PoS Tagging and Stemming for Agglutinative Languages.” ACM Transactions on Asian and Low-Resource Language Information Processing 18 (3): 1–21. https://doi.org/10.1145/3292398.
24. Brychcín, Tomáš, and Miloslav Konopík. 2015. “HPS: High Precision Stemmer.” Information Processing & Management 51 (1): 68–91. https://doi.org/10.1016/j.ipm.2014.08.006.
25. Bulletin, J Dawson - ALLC, and undefined 1974. n.d. “Suffix Removal and Word Conflation.”
26. Chen, Lin-Chih. 2018. “A Novel Page Clipping Search Engine Based on Page Discussion Topics.” Knowledge and Information Systems 58 (3): 525–50. https://doi.org/10.1007/s10115-018-1173-2.
27. Chen, Yan, Jie Wang, Ping Li, and Peilun Guo. 2019. “Single Document Keyword Extraction via Quantifying Higher-Order Structural Features of Word Co-Occurrence Graph.” Computer Speech & Language 57: 98–107. https://doi.org/10.1016/j.csl.2019.01.007.
28. Chintala, DR, EM Reddy - International Journal of Advanced Research In, and undefined 2013. n.d. “An Approach to Enhance the CPI Using Porter Stemming Algorithm.”
29. Cormen, Thomas H, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. 2009. *Introduction to Algorithms*. MIT press.
30. Deutsch, Avital, Hadas Velan, and Tamar Michaly. 2018. “Decomposition in a Non-Concatenated Morphological Structure Involves More than Just the Roots: Evidence from Fast Priming.” Quarterly Journal of Experimental Psychology 71 (1): 85–92. https://doi.org/10.1080/17470218.2016.1250788.
31. Djenouri, Youcef, Asma Belhadi, Djamel Djenouri, and Jerry Chun-Wei Lin. 2021. “Cluster-Based Information Retrieval Using Pattern Mining.” Applied Intelligence 51 (4): 1888–1903.
32. El Mahdaouy, Abdelkader, Éric Gaussier, and Saïd Ouatik El Alaoui. 2019. “Should One Use Term Proximity or Multi-Word Terms for Arabic Information Retrieval?” *Computer Speech & Language* 58: 76–97.
33. El-Defrawy, Mahmoud, Yasser El-Sonbaty, and Nahla A Belal. 2016. “A Rule-Based Subject-Correlated Arabic Stemmer.” Arabian Journal for Science and Engineering 41 (8): 2883–91. https://doi.org/10.1007/s13369-016-2029-2.
34. Frakes, William B, and Christopher J Fox. 2003. “Strength and Similarity of Affix Removal Stemming Algorithms.” ACM SIGIR Forum 37 (1): 26–30. https://doi.org/10.1145/945546.945548.
35. Gupta, Vaishali, Nisheeth Joshi, and Iti Mathur. 2013. “Rule Based Stemmer in Urdu.” *2013 4th International Conference on Computer and Communication* Technology *(ICCCT)*. IEEE. https://doi.org/10.1109/iccct.2013.6749615.
36. ———. 2015. “Design & Development of Rule Based Inflectional and Derivational Urdu Stemmer ‘Usal’” *2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE)*. IEEE. https://doi.org/10.1109/ablaze.2015.7154958.
37. Haq, Molvi Abdul. 1996. ( ), . .
38. Husain, Mohd Shahid, Faiyaz Ahamad, and Saba Khalid. 2013. “A Language Independent Approach to Develop Urdu Stemmer.” In *Advances in Intelligent Systems and Computing*, 178:45–53. Springer Verlag. https://doi.org/10.1007/978-3-642-31600-5_5.
39. Hussain, Sara. 2004. “Finite-State Morphological Analyzer for Urdu.” *Unpublished MS Thesis, Center for Research in Urdu Language Processing, National University of Computer and Emerging Sciences, Pakistan*.
40. Hussain, Z, S Iqbal, T Saba, AS Almazyad, & A Rehman - Journal of Theoretical, and undefined 2017. n.d. “DESIGN AND DEVELOPMENT OF DICTIONARY-BASED STEMMER FOR THE URDU LANGUAGE.”
41. Islam, Riaz Ahmed. 2012. “The Morphology of Loanwords in Urdu: The Persian, Arabic and English Strands.” PhD Thesis, Newcastle University.

42. Jabbar, Abdul, Sajid Iqbal, Adnan Akhunzada, and Qaisar Abbas. 2018. “An Improved Urdu Stemming Algorithm for Text Mining Based on Multi-Step Hybrid Approach.” Journal of Experimental & Theoretical Artificial Intelligence, 1–21. https://doi.org/10.1080/0952813x.2018.1467495.
43. Jabbar, Abdul, Sajid Iqbal, and Muhammad Usman Ghani Khan. 2016. “Analysis and Development of Resources for Urdu Text Stemming.” Language and Technology 1.
44. Jabbar, Abdul, Sajid Iqbal, Muhammad Usman Ghani Khan, and Shafiq Hussain. 2016. “A Survey on Urdu and Urdu like Language Stemmers and Stemming Techniques.” Artificial Intelligence Review 49 (3): 339–73. https://doi.org/10.1007/s10462-016-9527-1.
45. Jabbar, Abdul, Saif ul Islam, Shafiq Hussain, Adnan Akhunzada, and Manzoor Ilahi. 2019. “A Comparative Review of Urdu Stemmers: Approaches and Challenges.” Computer Science Review 34: 100195.
46. Kansal, Rohit, Vishal Goyal, and G S Lehal. 2012. “Rule Based Urdu Stemmer.” *Aclweb.Org*. https://www.aclweb.org/anthology/C12-3034.pdf.
47. Kaur, P, and PK Buttar. 2019. “A Rule-Based Stemmer for Punjabi Verbs.” http://www.academia.edu/download/60329350/IRJET-V6I5117420190818-76858-43x52e.pdf.
48. Khan, Sajjad Ahmad, Waqas Anwar, Usama Ijaz Bajwa, and Xuan Wang. 2012. “A Light Weight Stemmer for Urdu Language: A Scarce Resourced Language.” *Aclweb.Org*. http://www.the-comma.com/diacritics.php.
49. Khan, Sajjad, Waqas Anwar, Usama Bajwa, and Xuan Wang. 2015a. “Template Based Affix Stemmer for a Morphologically Rich Language.” 2. The International Arab Journal of Information Technology. Vol. 12.
50. ———. 2015b. “Template Based Affix Stemmer for a Morphologically Rich Language.” *International Arab Journal of Information Technology (IAJIT)* 12 (2).
51. Koirala, Pravesh, and Aman Shakya. 2020. “A Nepali Rule Based Stemmer and Its Performance on Different NLP Applications,” February. http://arxiv.org/abs/2002.09901.
52. Kumar, Ritesh, Guggilla Bhanodai, and Rajendra Pamula. 2019. “Book Search Using Social Information, User Profiles and Query Expansion with Pseudo Relevance Feedback.” Applied Intelligence 49 (6): 2178–2200.
53. Labani, Mahdieh, Parham Moradi, Fardin Ahmadizar, and Mahdi Jalili. 2018. “A Novel Multivariate Filter Method for Feature Selection in Text Classification Problems.” Engineering Applications of Artificial Intelligence 70: 25–37. https://doi.org/10.1016/j.engappai.2017.12.014.
54. Lovins, Julie Beth. 1968. “Development of a Stemming Algorithm.” Mech. Transl. Comput. Linguistics 11 (1–2): 22–31.
55. Mehmood, Khawar, Daryl Essam, Kamran Shafi, and Muhammad Kamran Malik. 2019. “Sentiment Analysis for a Resource Poor Language—Roman Urdu.” ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP) 19 (1): 1–15.
56. Meitei, S Poireiton, Bipul Syam Purkayastha, and H Mamata Devi. 2015. “Development of a Manipuri Stemmer: A Hybrid Approach.” *2015 International Symposium on Advanced Computing and Communication (ISACC)*. IEEE. https://doi.org/10.1109/isacc.2015.7377328.
57. Momenipour, F, MR Keyvanpour - International Journal Of, and undefined 2016. n.d. “PHMM: Stemming on Persian Texts Using Statistical Stemmer Based on Hidden Markov Model.” *Search.Ebscohost.Com*. http://search.ebscohost.com/login.aspx?direct=true{\&}profile=ehost{\&}scope=site{\&}authtype=crawler{\&}jrnl=20088302{\&}AN=117514060{\&}h=M9N0fPg{\%}2FioFnlFo4wRW6ESzNrY1{\%}
58. 2FtH40n662LeMI2PjSGQh75jQNrvgK51hPWw0VVzN0XZZ0JI50hzkj0GtJ{\%}2Bg{\%}3D{\%}3D{\&}crl=c.
59. Mustafa, Arazo M, and Tarik A Rashid. 2017. “Kurdish Stemmer Pre-Processing Steps for Improving Information Retrieval.” Journal of Information Science 44 (1): 15–27. https://doi.org/10.1177/0165551516683617.
60. Nguyen, Hien T, Phuc H Duong, and Erik Cambria. 2019. “Learning Short-Text Semantic Similarity with Word Embeddings and External Knowledge Sources.” Knowledge-Based Systems 182: 104842. https://doi.org/10.1016/j.knosys.2019.07.013.
61. Paice, Chris D. 1994. “An Evaluation Method for Stemming Algorithms.” *SIGIR ’94*. Springer London. https://doi.org/10.1007/978-1-4471-2099-5_5.
62. Paik, Jiaul H, and Swapan K Parui. 2011. “A Fast Corpus-Based Stemmer.” ACM Transactions on Asian Language Information Processing (TALIP) 10 (2): 1–16.
63. Parveen, A. 2014. “Morphological Analysis of Modern Standard Urdu.” http://hdl.handle.net/10603/52303.
64. Patil, CG, SS Patil - Journal of Electronics, Communication and Soft, and undefined 2013. n.d. “Use of Porter Stemming Algorithm and SVM for Emotion Extraction from News Headlines.” *Ijecscse.Org*. http://www.ijecscse.org/papers/June2013/Use of Porter Stemming Algorithm and SVM for Emotion Extraction from News Headlines.pdf.
65. Porter, M F. 1980. “An Algorithm for Suffix Stripping.” Program 14 (3): 130–37. https://doi.org/10.1108/eb046814.
66. Qureshi, AH, DB Anwar, M Awan - International Journal of Research in Linguistics And, and undefined 2012. n.d. “Morphology of the Urdu Language.”
67. Rahimi, Adel. 2015. “A New Hybrid Stemming Algorithm for Persian,” July. http://arxiv.org/abs/1507.03077.
68. Razali, Ansari, Salwani Mohd, Nor Azan, and Faezehsadat Shahidi. 2020. “Stemming Text-Based Web Page Classification Using Machine Learning Algorithms: A Comparison.” International Journal of Advanced Computer Science and Applications 11 (1). https://doi.org/10.14569/ijacsa.2020.0110171.
69. Rizvi, S M Jafar, and Mutawarra Hussain. 2005. “Analysis, Design and Implementation of Urdu Morphological Analyzer.” *2005 Student Conference on Engineering Sciences and Technology*. IEEE. https://doi.org/10.1109/sconest.2005.4382901.
70. Rouibia, Rima, Imane Belhadj, and Mohamed Amine Cheragui. 2017. “JIDR: Towards Building Hybrid Arabic Stemmer.” *2017 International Conference on Mathematics and Information* Technology *(ICMIT)*. IEEE. https://doi.org/10.1109/mathit.2017.8259714.
71. Saeed, Ari M, Tarik A Rashid, Arazo M Mustafa, Rawan A Al-Rashid Agha, Ahmed S Shamsaldin, and Nawzad K Al-Salihi. 2018. “An Evaluation of Reber Stemmer with Longest Match Stemmer Technique in Kurdish Sorani Text Classification.” Iran Journal of Computer Science 1 (2): 99–107.

https://doi.org/10.1007/s42044-018-0007-4.

72. Savoy, Jacques. 1993. “Stemming of French Words Based on Grammatical Categories.” Journal of the American Society for Information Science 44 (1): 1–9. https://doi.org/10.1002/(sici)1097-4571(199301)44:1<1::aid-asi1>3.0.co;2-1.
73. Schmidt, Ruth Lail. n.d. *URDU: AN ESSENTIAL GRAMMER*. First. Routledge.
74. SHAH, MR, H Shaikh, JA MAHAR, and SA MAHAR. 2016. “Sindhi Stemmer for Information Retrieval System Using Rule-Based Stripping Approach.” Sindh University Research Journal-SURJ (Science Series) 48 (4).
75. Sirsat, Sandeep R, Vinay Chavan, and Hemant S Mahalle. 2013. “Strength and Accuracy Analysis of Affix Removal Stemming Algorithms.” International Journal of Computer Science and Information Technologies 4 (2): 265–69.
76. Suryani, Arie Ardiyanti, Dwi Hendratmo Widyantoro, Ayu Purwarianti, and Yayat Sudaryat. 2018. “The Rule-Based Sundanese Stemmer.” ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP) 17 (4): 1–28.
77. Taghi-Zadeh, Hossein, Mohammad Hadi Sadreddini, Mohammad Hasan Diyanati, and Amir Hossein Rasekh. 2015. “A New Hybrid Stemming Method for Persian Language.” Digital Scholarship in the Humanities, fqv053. https://doi.org/10.1093/llc/fqv053.
78. Tubishat, Mohammad, Mohammad AM Abushariah, Norisma Idris, and Ibrahim Aljarah. 2019. “Improved Whale Optimization Algorithm for Feature Selection in Arabic Sentiment Analysis.” Applied Intelligence 49 (5): 1688–1707.
79. UEP. 2014. <background-color:#FFD9B3;idirection:rtl;> </background-color:#FFD9B3;idirection:rtl;>. Unique Education Publisher, Urdu bazar Lahore.
80. W1. n.d. “Retrieved from Http://Www.Bbc.Com/Urdu.” W1. Retrieved from http://www.bbc.com/urdu.
81. W2. n.d. “Retrieved from Https://Www.Dawnnews.Tv.”
82. W3. n.d.. “. Retrieved from Http://Www.Urduencyclopedia.Org/Urdudictionary.”
83. W4. n.d. “Retrieved from Http://Www.Cle.Org.Pk/Software/Ling_resources/Wordlist.Htm.”
84. W5. n.d. “Retrieved from Http://Cle.Org.Pk/Software/Ling_resources/UrduHighFreqWords.Htm.”
85. Yusuf, Nuhu, Mohd Amin Mohd Yunus, and Norfaradilla Wahid. 2020. “Arabic Text Stemming Using Query Expansion Method.” In *Advances in Intelligent Systems and Computing*, 1073:3–11. Springer. https://doi.org/10.1007/978-3-030-33582-3_1.
86. Zeroual, Imad, Mohamed Boudchiche, Azzeddine Mazroui, and Abdelhak Lakhouaja. 2017. “Developing and Performance Evaluation of a New Arabic Heavy/Light Stemmer.” *Proceedings of the 2nd International Conference on Big Data, Cloud and Applications*. ACM. https://doi.org/10.1145/3090354.3090371.

Figures

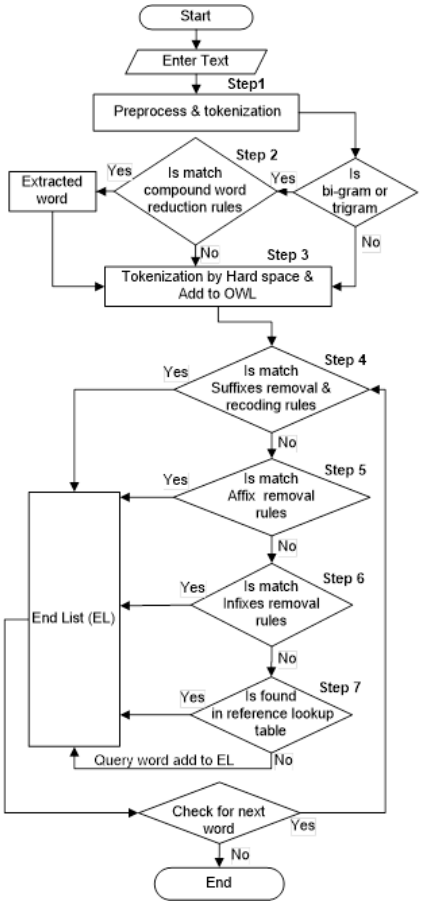

Figure 1

Proposed framework for UTS

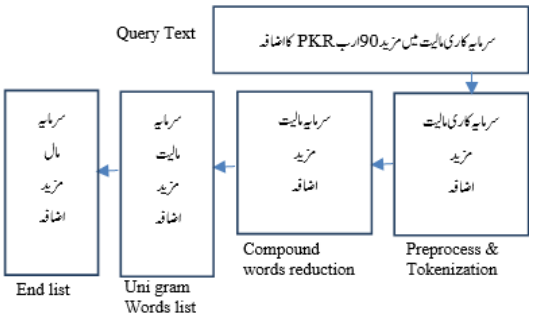

Figure 2

Example of tokenization process

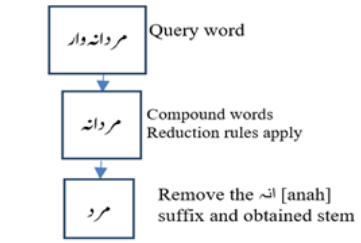

Figure 3

Example of compound words reduction rules

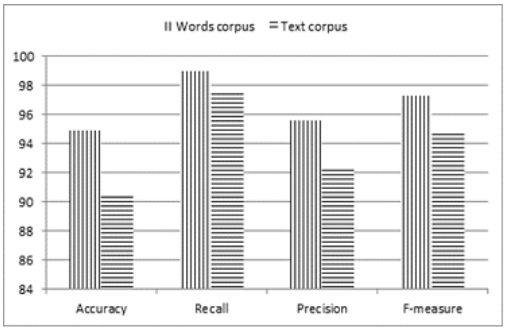

Figure 4

Result comparison on word corpus and text corpus

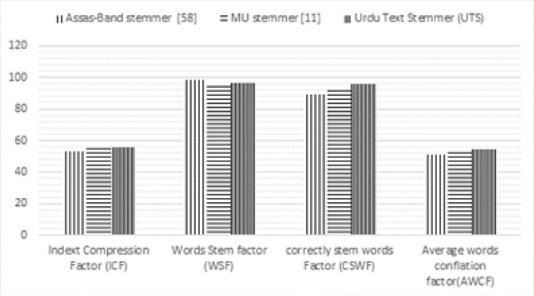

Figure 5

Results comparison state of arts Urdu stemmer on standard dataset

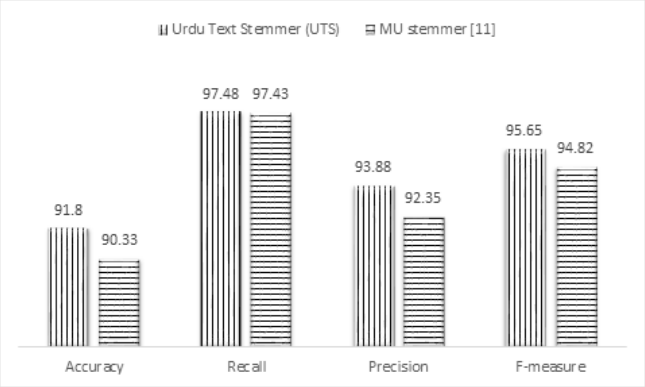

Figure 6

Result comparison proposed stemmer with MU Stemmer

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- APPENDIX.docx