American Journal of Computing Research Repository, 2014, Vol. 2, No. 1, 1-7
Available online at http://pubs.sciepub.com/ajcrr/2/1/1
© Science and Education Publishing
DOI:10.12691/ajcrr-2-1-1

Word Segmentation Model for Sindhi Text

Zeeshan Bhatti*, Imdad Ali Ismaili, Waseem Javaid Soomro, Dil Nawaz Hakro

Institute of Information and Communication Technology, University of Sindh, Jamshoro
*Corresponding author: zeeshan.bhatti@usindh.edu.pk

Received November 25, 2013; Revised December 15, 2013; Accepted January 01, 2014

Abstract Through this research the problem of Sindhi Word Segmentation has been addressed and various techniques have been discussed to solve this problem. Word Segmentation is the preliminary phase involved in any tool based on Natural Language Processing (NLP). For any system to understand the written text, it needs to be able to break it into individual tokens for processing. Sindhi being a cursive ligature based Persio-Arabic script, is quite complex and rich having large number of characters in its script with all characters having multiple glyph's based on its position in the text. In this paper Sindhi word Tokenization model has been proposed implementing various algorithms showing the process of tokenizing Sindhi text into individual words for corpus building and creating word repository for Sindhi Spell, grammar checker and other NLP applications. The problem of tokenization is resolved by first identifying the sentence boundaries and extracting each sentence into isolated list form, where each list element is a complete sentence. Then the segregated sentences are broken down into words with hard space character used as word boundaries and soft spaces are considered as part of word and thus ignored from segmenting. Finally each word is again filtered to remove special characters and then each word is converted and saved as token after validation.

Keywords: *word segmentation, sindhi tokenization, sindhi language, Sindhi Spell Checker*

Cite This Article: Zeeshan Bhatti, Imdad Ali Ismaili, Waseem Javaid Soomro, and Dil Nawaz Hakro, "Word Segmentation Model for Sindhi Text." *American Journal of Computing Research Repository* 2, no. 1 (2014): 1-7. doi: 10.12691/ajcrr-2-1-1.

1. Introduction

The process of segregating and isolating the sentence into individual token of words, is termed as Word segmentation or tokenization [1]. In Natural Language Processing (NLP) the term tokenization or word segmentation is deemed as the most fundamental task [2]. Almost every application of NLP requires at certain stages the process of breaking its text into individual tokens for processing -for example, in Machine Translation (MT) and Spell Checking [2,3]. The tokenization process is done by identifying word boundaries in languages like English where punctuation marks or white spaces are used to segregate words [3]. The scanning routines usually include various algorithms for handling morphology in a language-dependent manner. Even for a language like English, which is very lightly inflected, the phenomena of contraction and possessives will also need to be handled within the word extraction routines [4,5]. Sindhi, similar to other Asian languages -like Urdu, Arabic, Persian, endures the same problem of text segmentation with space omission and insertion issues.

Sindhi is an official State language of Sindh province in Pakistan and is spoken by approximately 34.4 million people in Pakistan and around 2.8 million people in India [6]. Sindhi script is based on Persio-Arabic script, with Arabic Nashk style of writing, from Right-to-Left direction with cursive ligature system [6]. Sindhi script has cursive behavior in its written form, having subsequent characters; in a word, joined with each other as shown in Figure 1. Due to its cursive nature and having Aerabs (diacritics marks) makes Sindhi text difficult to process in applications of NLP. For any application of NLP it's extremely vital that a standard corpus of a language is built so that the text can be processed and compared with some statistical analysis [7]. Therefore, the need for developing a formal Sindhi corpus is eminent and a model is needed for the tokenization of Sindhi words.

This paper discusses the Sindhi word segmentation technique for the development of Sindhi corpus and tokenizing Sindhi text, to build a repository for Sindhi words for NLP applications like Spell Checkers. Sindhi word boundaries from within the text are identified by finding the hard space character. Sindhi; being a very complex language, possess fifty two characters in its script with each character having separate glyph shapes, based on the position of each character in a string. This consequently generates the case of ambiguity in Sindh in Script, as the Sindhi language contains two types of letters – connectors and non-connectors. Sindhi word therefore uses soft space as well as hard space characters as shown in Figure 1.

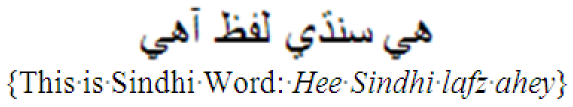

هي سنڌي لفظ آهي
{This·is·Sindhi·Word:·*Hee·Sindhi·lafz·ahey*}

Figure 1. Sindhi text

1.1. Related Work

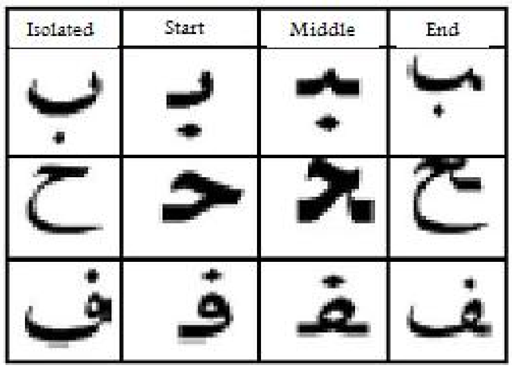
Most of the modern languages in the world have already developed various tools and techniques for segmenting their written text and documents for spell checking and correction. A part from languages of the European countries, the algorithms for word tokenization has been implemented for various other languages spoken in Asian counties. Some of the relevant work done in this regard includes: word segmentation done for Arabic language [8,9,10,11], Bangla [12], Hindi [13], Nepali [14], Tamil [15] and Urdu [16,17]. These are few examples; however, unfortunately very little work has been done in this regard for Sindhi Language.

For segmenting Arabic text, Sheikh et al, proposes Arabic Words/sub-words segmentation into characters using primary and secondary strokes with vertical projection graphs [18] but for OCR systems only and not for digital text. Similarly Shaikh et al. uses Height Profile Vector (HPV) to segment Sindhi Characters [19], but again for printed or handwritten scanned Sindhi text. His approach addresses the problem of segmenting for OCR systems and not for digital text. Durrani N. and Hussain S. address the orthographic and linguistics features of Urdu language for word segmentation, employing a hybrid solution of n-gram ranking with rule based matching heuristics [3]. On the other hand, Akram M. in his thesis discusses statistical solution of word segmentation for Urdu Language [20] but again for OCR systems. However, Mahar et al. develops five Algorithms based on Lexicon driven approach for Sindhi Word Segmentation into possible morpheme sequences [21].

Similarly the most relevant work on Sindhi Text Segmentation is done by Mahar et al. discussed in [1], in which he presents a layer based model for Sindhi text segmentation. However in his work he uses three layers, where each layer segment words with varying degree of intricacy, from simple, compound to complex Sindhi words. Contrary to this and other techniques discussed, we have addressed the problem of segmenting Sindhi Words that are already in digital or textual form, taken from internet or typed into a word processor for the purpose of corpus building, constructing word repository, machine translation, spell checking, grammar checking and text to speech systems etc. Our adapted technique works on identifying sentence boundaries, then tokenizing the words from sentence and validating the isolated word for accuracy.

1.2. Character Glyph and Space Types in Sindhi Text

The 52 characters in Sindhi script have multiple shape or glyph representations according to their position in the word. There are four different category of shapes that a character may possess with respect to its placement in text, initial or start, medial or middle, final or end, and standalone or isolated as shown in Figure 2. Soft spaces in Sindhi script are used to separate certain characters that do not have cursive context sensitive shapes for all four positions. For example in Table 1 the character {Dhaal} 'ڏ' has two basic types of cursive shapes, isolated and start shape, and middle and end shape. When ڏ is used at start of the word, it does not join with any other characters. Hence, a soft space is inserted to indicate the separation. This soft space does not indicate the word boundary as it is not a character, as compared to hard space, which is a character itself having no glyph or shape and occupies space.



Isolated
Start
Middle
End

Figure 2. Different shapes of characters according to position in a Word

Table 1. Shape group of 'ڏ' character

Isolate	Start	Middle	End
ذ	ذرا	لذني	لذني

As the hard space is an individual character having a specific ASCII (32) and Unicode (u+0020) designated code that occupies memory and screen space when used, provide a very easy to identify marker for word boundaries in segmentation process. Whereas the use of soft spaces is majorly to do with the character shape groups and placement, thus it does not possess any form of ASCII or Unicode representation, nor does it occupy memory. Therefore, the soft space is not considered as an individual character and consequently it is not used to identify word boundaries.

2. Architecture of the System

The System under study has been segregated into two main sections. In the first section, segmenting Sindhi words into tokens is performed and in the second section, verifying each generated tokens is done. The validation of results is done by using a prebuilt words repository. Previously developed Sindhi word processor software (by the author [6]) is used as the primary tool for working with Sindhi text and showing the results. In the first section, set of algorithms and routines have been implemented to scan the text and extract the Sindhi words and then each word is compared to a repository of Sindhi words for verification. The scanning routines evaluate each word using a set of rules to be a valid token or not. If a token is found to be invalid, then it is marked as incorrect and is simply casted away by putting it inside a list with all ignored and unwanted words. This development methodology is illustrated in Figure 3 showing various stages of the system. These stages of the system architecture are explained in the sub sections below.

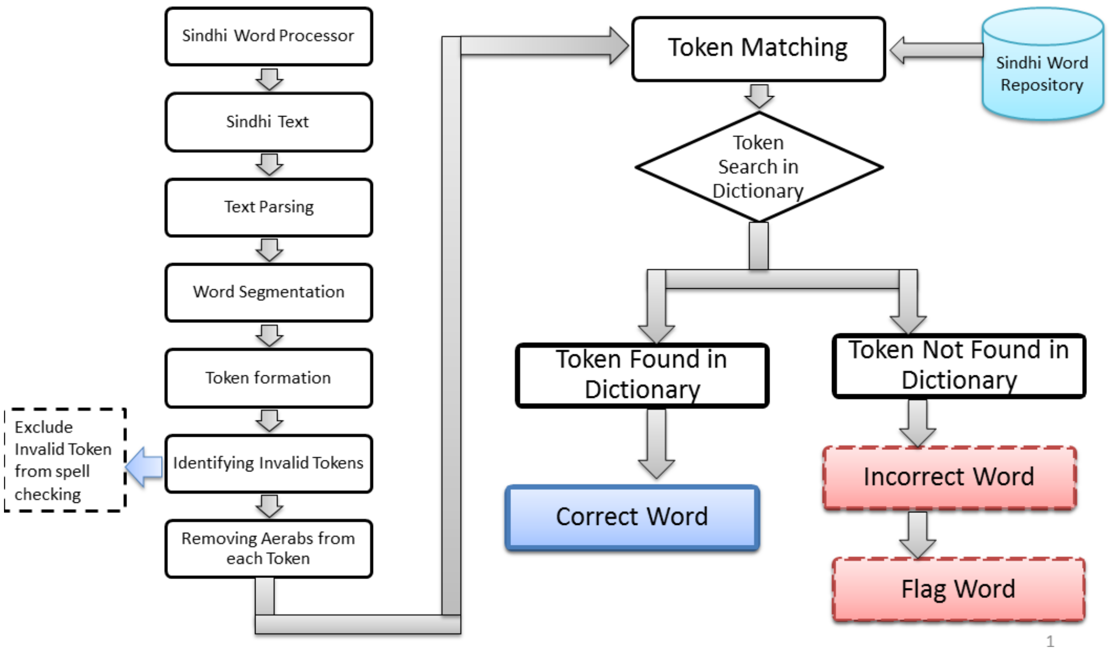

Sindhi Word Processor
Sindhi Text
Text Parsing
Word Segmentation
Token formation
Identifying Invalid Tokens
Exclude Invalid Token from spell checking
Removing Aerabs from each Token
Token Matching
Sindhi Word Repository
Token Search in Dictionary
Token Found in Dictionary
Token Not Found in Dictionary
Correct Word
Incorrect Word
Flag Word

Figure 3. Various development stages of the System

3. Scanning and Extraction of Sindhi Words

The first stage of developing Sindhi word segmentation model, works at four different levels. The system is able to do:

1. Text segmentation into sentences.
2. Sentence Segmentation into Words.
3. Tokens Creation.
4. Token Matching.

At each level, set of routines are used to parse Sindhi text, extracting sentences and creating valid tokens. These routines are discussed further in the subsequent sections.

3.1. Text Segmentation into Sentences

Initially Sindhi text written in Sindhi Word Processor document is read and scanned. Then the text is separated into sentences by identifying the sentence boundaries which is from the start of the text to the next full stop (.) as shown in Figure 4 below. The sentence boundaries are identified by a full stop (.) and a question mark (?). Along with these two, sentence boundaries are also marked by identifying the end of paragraph and start of new line.

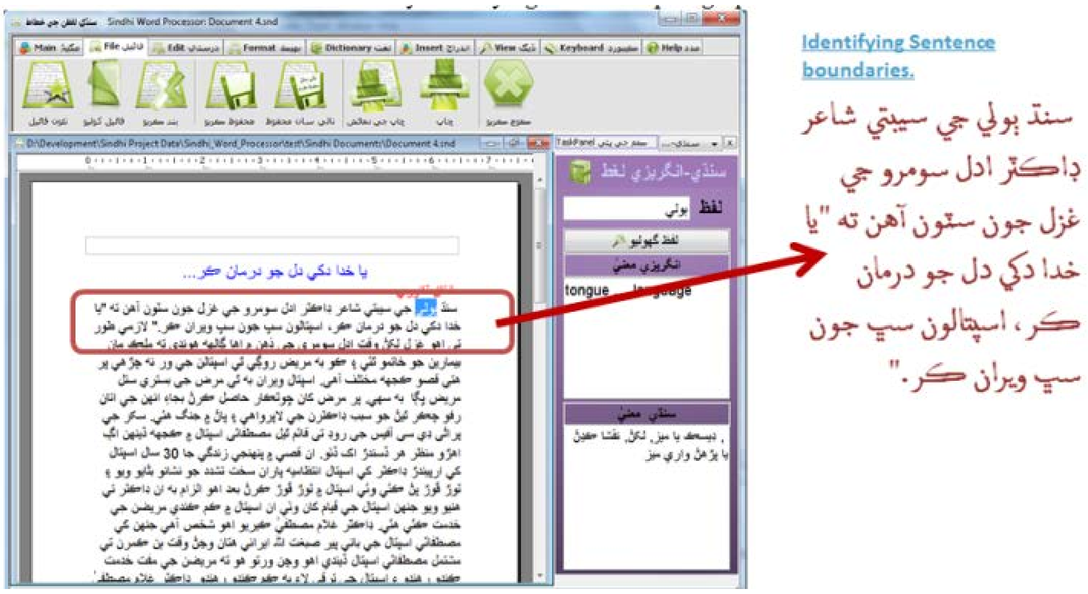

Sindhi Word Processor: Document 4.snd
Main
File
Edit
Format
Dictionary
Insert
View
Keyboard
Help
Identifying Sentence boundaries.
سنڌ يولي جي سيتي شاعر ڊاڪٽر ادل سومرو جي غزل جون سٽون آهن ته "يا خدا دکي دل جو درمان ڪر، اسپتالون سپ جون سپ ويران ڪر."

Figure 4. Scanning and identifying sentence boundaries

Each input text is scanned first from the beginning of the text starting from the initial starting position of the sentence. Then end of sentence identifier, which is either a full stop (.) or a question mark (?), is searched for and marked. The system then fetches the sentence form the start index to the end index. After the first sentence is isolated the next sentence is searched for and the start and end index markers are reset. Following is the implementation details for scanning Sindhi text and identifying sentence boundaries.

Algorithm 1: Code for scanning Sindhi text and extracting the sentence by identifying its boundaries

```
BEGIN Synchronized PROCEDURE sindhiSentenceIdentifier (ARGUMENT text Document)
DECLARE int indexStart
DECLARE int indexEnd
DECLARE sindhiSentenceList
//Exception Handling
TRY BEGIN
        ASSIGN sindhiText = textDocument.getText()
        ASSIGN indexStart = textDocument.getCaretStartPostion()
        FOR indexEnd <= sindhiText.getLength() THEN
                IF sindhiText.getLength() != 0 THEN
                        END PROCEDURE
                ELSE
                        IF sindhiText.indexOf(".") != -1 THEN
                              ASSIGN indexEnd = textDocument.indexOf(".")
                        IF sindhiText.indexOf("?") != -1 THEN
                              ASSIGN indexEnd = textDocument.indexOf("?")
                        IF sindhiText.indexOf("\r") != -1 THEN
                              ASSIGN indexEnd = textDocument.indexOf("\r")
                        IF sindhiText.indexOf("\n") != -1 THEN
                              ASSIGN indexEnd = textDocument.indexOf("\n")
                END ELSE

                ASSIGN sindhiSentenceList.ADD sindhiText.substring(indexStart, indexEnd)
                ASSIGN temp = indexStart
                ASSIGN indexStart = indexEnd+1

                END FOR
    END TRY
 CATCH Exception
PRINT ERROR Message
END PROCEDURE
```

3.2. Sentence Segmentation into Words

For the segmentation and creation of the Sindhi word from sentences, word boundaries have to be identified and marked. The word boundaries are determined by the space character before and after each word. As discussed earlier, Sindhi script possess two types of space characters in written or typed form, hard spaces and soft space. For the purpose of word token generation the hard space has been used in this system as a word boundary identifier. The soft spaces are ignored and counted as part of word as shown in Figure 5.

يا خدا دکي دل جو درمان ڪر، اسپتالون سپ جون سپ ويران ڪر
^ ^ ^ ^ ^ ^ ^^ ^ ^ ^ ^ ^ ^

{O-Allah heal the torment mourning hearts, make the hospitals an uninhabited place: *Ya Khudda dukhey dil joo darman ker, aspetaloon saab jun saab weraan ker*}

Figure 5. Sindhi text with word boundaries marked at hard spaces

Here it is to be noted that in our system we have adopted and used the terminology of token and word separately. We consider the word to be in a general form separated using spaces. The generated word token may be a misspelled or incorrectly typed word having invalid spelling. The word may also contain characters such as punctuation marks, special symbols such as: "@, &, *, !, #, etc.". Thus at this point all such words are considered to be a general form of words correctly segmented and segregated. Whereas, the tokens generated will be correctly spelled words verified from a correctly spelled repository of Sindhi Words. At the beginning of process we change the local environment variable to Arabic 'AR' for our compiler and interpreter to be able to read and process the text from right-to-left order for Sindhi script. Then, a break iterator object, which is used to break each word according to the index of hard space, is declared. The word is isolated for further verification and validation before it can be saved into a token. This process of segmentation differentiates between words and characters that are not part of words. These characters are ignored and skipped to achieve accuracy of tokens formed. The ignored characters include spaces, tabs, punctuation marks, and most symbols, have word boundaries on both sides. In algorithm 2 the implementation details have been provided for segmenting sentences into words.

Algorithm 2: Code for segmenting words and creating word tokens for further process.

```
BEGIN synchronized PROCEDURE getSindhiWordBoundary (ARGUMENT sindhiText) {
DECLARE and SET Locale currentLocale = new Locale ("ar","AR");
SET BreakIterator wordIterator = currentLocal
CALL markBoundaries(sindhiText, wordIterator);
END PROCEDURE

BEGIN PROCEDURE markBoundaries(ARGUMENT sindhiText, ARGUMENT breakIterator) {
        DECLARE StringBuffer markers
        SET markers.setLength(sindhiText.length() + 1);
        FOR (k = 0; k < markers.length(); k++) BEGIN
                markers.setCharAt(k,' ');
        END FOR
        SET breakIterator.setText(sindhiText);
        DECLARE boundary = breakIterator.first();
       WHILE (boundary != BreakIterator.DONE) BEGIN
                SET markers.setCharAt(boundary,' ');
                SET boundary = iterator.next();
       END WHILE
       DECLARE HashList
       PUT markedWords INTO HashList
END PROCEDURE
```

3.3. Sindhi Word Tokens

After the identification of word boundaries, each word is then isolated and put inside a hash list. The words are identified and each word from the list is retrieved and analyzed for validity. Each token is created by identifying the correctly spelled word and removing any additional unnecessary characters that may be part of the original words. This filtration process involves traversing each character with the word and removing all special characters such as @,#,$,%,^,&,*,(,),_,+,-,=,{,},|,[,],:,;,<,>,?, etc. All these type of characters can be part of the string and may have been attached to word in previous stage. Along with these, the hard space characters, newline and new paragraph symbols are also trimmed out. More importantly the filtration process eradicates the occurrence of any letter from English alphabet as it is very common to use English words at certain places in a Sindhi document or article. In the last stage, each word is compared form a known list of repository of Sindhi Words for final validation. Figure 6 shows the tokens created.

يا	{ya :Oh }
خدا	{Khuda :Allah}
دکي	{dukhey:Mournig}
دل	{dil:Heart}
جو	{joo :the}
درمان	{darman:heal}
ڪر	{ker:do}
اسپتالون	{Aspetaloon:Hospitals}
سپ	{saab:all}
جون	{jun :Of}
سپ	{saab:all}
ويران	{weraan:uninhapitent}
ڪر	{ker :make}

Figure 6. Word tokens created from a sentence

In the following algorithm 3, the implementation of token creation has been shown. Here 'filter Char' array contains the information to filter the invalid characters from the word. For Sindhi word segmentation system invalid character are those characters that are not in Sindhi alphabet, such as punctuation marks, extra spaces, numerical, braces, etc.

Algorithm 3: Code for creating word tokens and filtering invalid tokens.

```
1.  BEGIN synchronized PROCEDURE filterSindhiTokens
2.  GET hashList of Sindhi Words
3.  DECLARE Enumeration   enum
4.  DECLARE filterChar[]<char Array>=
    {\r\n\t\f\"\\`~!@#$%^&*()_+-={}|[]:;<>?,./‘      “+
    “\u201C\u201D\u2018\u2019\b\205\u2028\u2029};

5.  PUT hashList INTO enum
6.  FOR enum.hasMoreElements() THEN
7.          ASSIGN String word = enum.nextElement()
8.          CONVERT word into Char_array [] <char>
9.          FOR index<=Char_array.length THEN
10.                 IF  Char_array[index]  ==  filterChar[f_index]
    THEN
11.                         SKIP this Character
12.                 ELSE
13.                         APPEND Char_array[index] INTO
    newWord<String>
14.             INCREMENT index
15.             INCREMENT f_index
16.         END FOR
17.         PUT newWord<String>   INTO TokenList<HashList>
18. END FOR
```

3.4. Token Matching

After creating each token from the Sindhi text, the tokens are searched and matched with the Sindhi wordlist from the repository. The system uses hash tables to store the wordlist hence the basic operations of searching and matching become very simple. We have used the same technique of matching has key to search a token from the hash table based repository as discussed in [22].The system uses Hash Structure Algorithm for fast and efficient searching. The analysis and verification of words also involves validating the error patterns and trends in spelling mistakes that occurs while typing Sindhi text, results of which have already been published in [23]. The Figure 7 shows the basic structure of the token matching done by the system.

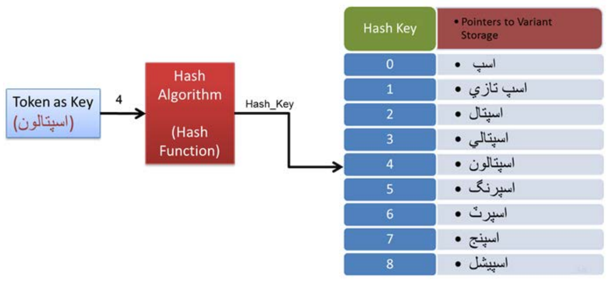


Figure 7. A Hash Table Structure for Sindhi Words

4. Results

The proposed model has been tested on various corpus of Sindhi text collected through internet (general articles and news articles) and from publisher (book chapter and Digital dictionary). The detail tokenization report is shown in Table 2. The articles taken form Sindhi literature books, were initially typed into the Sindhi word processor with a Sindhi keyboard designed for Sindhi typing used in [24]. Total of 157,509 words were generated by the proposed tokenization model, among which 146,132 words were verified and marked as correct tokens and valid Sindhi word tokens having the cumulative accuracy 92.78% of the model. Some 4645 tokens that were generated were considered invalid by the system and 7732 tokens were completely ignored due to some anomalies like having special characters or unknown Unicode literal in them.

Table 2. Results of the proposed segmentation model

Source	Paragraphs	Lines	Words	Incorrect Tokens	Tokens ignored	Total Words
15 Articles	14	1189	13622	1877	1332	16831
100 News Articles	224	2363	30267	1534	745	32546
Book Chapters (5)	104	1055	19589	1234	1689	22512
Sindhi Digital Dictionary	---	---	82654	---	2966	85620
Total			1,46,132	4645	7732	1,57,509

The Figure 8 (a) below shows the diagrammatic comparison among the tokens generated from various corpus of Sindhi Script. The Figure 8 (b) shows the cumulative accuracy compares to the incorrect and ignored tokens by the model.

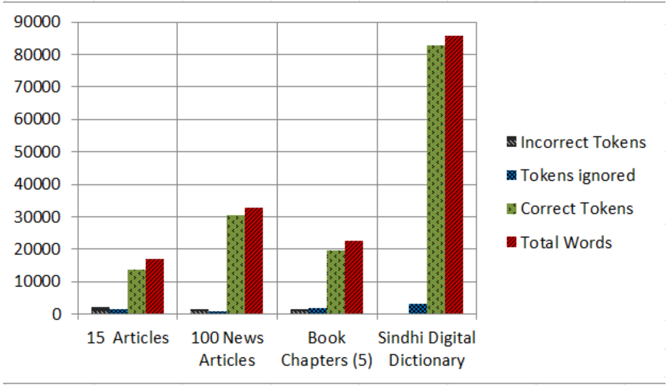


Figure 8(a). Bar graph showing the tokens generated

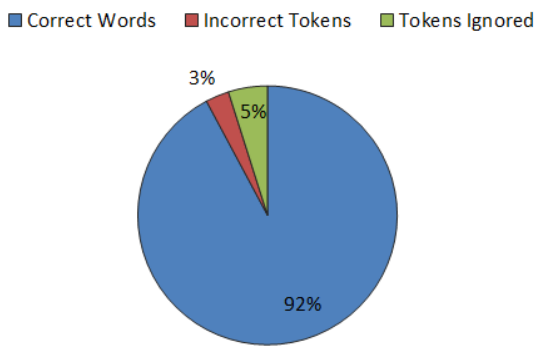


Figure 8(b). Pie chart showing the cumulativeaccuracy of the system

The overall accuracy of generating tokens by the proposed word segmentation model is shown given in Table 3 with the graph shown in Figure 9 illustrating the accuracy difference between various corpus of Sindhi text and how tokenization is varied in them.

Table 3. Accuracy of Proposed Model

Source	Accuracy %
15 Articles	80.94
100 News Articles	93
Book Chapters (5)	87.02
Sindhi Digital Dictionary	96.54

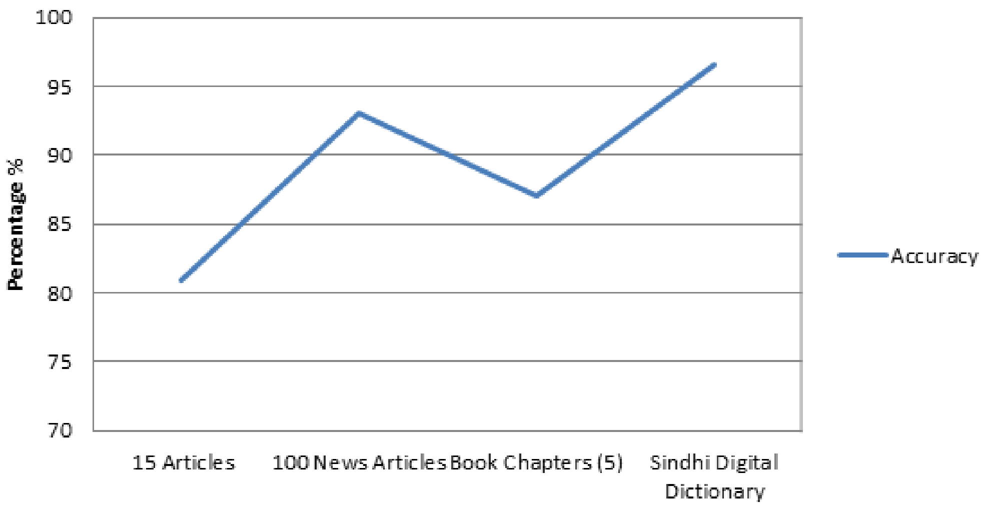


Figure 9. Graphical representation of calculated accuracy in proposed word segmentation model

5. Conclusion

The work presented in this paper shows the technique and algorithms used to tokenize Sindhi words from a given Sindhi text document. Each algorithm has been discussed and implementation given. The results are analyzed by checking each token generated by the system with a given list of words from a prebuilt Sindhi words repository (for spell checking). Each token is identified by the system to be correct and all invalid token are marked as an incorrect and are ignored. The results of proposed model are very sizable and accurate. The algorithm can be further utilized to segment Sindhi words for various other NLP purpose like machine translation, spell checking, grammar checking and text to speech systems.

References

[1] Mahar, J. A., Shaikh, H., Memon,G. Q., "A Model for Sindhi Text Segmentation into Word Tokens", Sindh University Research Journal (Science Series), Vol.44 (1) pp.43-48 (2012).

[2] Haruechaiyasak, C.; Kongyoung, S.; Dailey, M.; "A comparative study on Thai word segmentation approaches," Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2008. ECTI-CON 2008. 5th International Conference on, vol.1, no., pp.125-128, 14-17 May 2008.

[3] Nadir D. And Sarmad H. 2010. Urdu word segmentation. In Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT '10). Association for Computational Linguistics, Stroudsburg, PA, USA, 528-536.

[4] "A fast morphological algorithm with unknown word guessing induced by a dictionary for web search engine" Source: http://company.yandex.ru/articles/iseg-las-vegas.xml Retrieved on: 12, June 2011.

[5] Hull, D. A, "Stemming Algorithms A Case Study for Detailed Evaluation," (Rank Xerox Research Centre), JASIS vol. 47, 1996.

[6] Ismaili, I.A, Bhatti, Z., Shah, A. A. "Design and Development of Graphical User Interface for Sindhi Language (GUISL)". Mehran University Research Journal of Engineering & Technology, Volume 30, No. 4, October 2011.

[7] Rahman M U (2010). Towards Sindhi Corpus Construction, Conference on Language and Technology, Lahore, Pakistan.

[8] Shaalan K. "Arabic GramCheck: A Grammar Checker for Arabic", Software Practice and Experience, John Wiley & sons Ltd., UK, 35(7):643-665, June 2005.

[9] Zribi, C. B. O. And Ben Ahmed, M. 2003. "Efficient automatic correction of misspelled Arabic words based on contextual information." In Proceedings of the 7th International Conference on Knowledge-Based Intelligent Information and Engineering Systems (KES'03). V. Palade, R. J. Howlett, and L. Jain Eds., Oxford, Springer, 770-777.

[10] Farghaly, A., Shaalan, K. "Arabic Natural Language Processing: Challenges and Solutions," ACM Transactions on Asian Language Information Processing (TALIP), the Association for Computing Machinery (ACM), 8(4)1-22, December 2009, 8(4), 1-22.

[11] Shaalan, K., Allam, A., Gohah, A., "Towards Automatic Spell Checking for Arabic". Conference on Language Engineering, ELSE, Cairo, Egypt, 2003.

[12] Uzzaman, N., And Khan, M., "A Double Metaphone Encoding for Bangla and its Application in Spelling Checker", Proc. 2005 IEEE Natural Language Processing and Knowledge Engineering, Wuhan, China, October, 2005.

[13] Chaudhuri,B. B., "Towards Indian Language Spell-checker Design," lec, pp.139, Language Engineering Conference (LEC'02), 2002.

[14] Bal K. B. Et. Al., "Nepali Spellchecker", PAN Localization Working Papers 2004-2007, Centre for Research in Urdu Language Processing, National University of compute and Emerging Sciences, Lahore, Pakistan, pp.316-318.

[15] Dhanabalan, T., Parthasarathi, R., & Geetha, T. V. (N.D.). "Tamil Spell Checker" Resource Center for Indian Language Technology Solutions – Tamil, School of Computer Science and Engineering, Anna University, Chennai, India, pp.18-27. 2003.

[16] Naseem, T., & Hussain, S. "Spelling Error Corrections for Urdu". Published online: 26 September 2007 © Springer Science Business Media B.V. 2007. PAN Localization Working Papers 2007, Centre for Research in Urdu Language Processing, National University of compute and Emerging Sciences, Lahore, Pakistan, pp.117-128.

[17] Naseem, T. And Hussain, S, "A Novel Approach for Ranking Spelling Mistakes in Urdu", Language Resources and Evaluation, 2007. 41:117-128.

[18] Shaikh, N. A., Shaikh, Z. A., & Ali, G. (2009). Segmentation of Arabic text into characters for recognition. In Wireless Networks, Information Processing and Systems (pp.11-18). Springer Berlin Heidelberg.

[19] Shaikh, N. A., Mallah, G. A., & Shaikh, Z. A. (2009). Character Segmentation of Sindhi, an Arabic Style Scripting Language, using Height Profile Vector.Australian Journal of Basic and Applied Sciences, 3(4), 4160-4169.

[20] Akram, M. (2009) “Word Segmentation for Urdu OCR System”, Master’s Thesis, Department of Computer Science, National University Of Computer & Emerging Sciences, Lahore, Pakistan.

[21] Mahar, J.A., Memon, G. Q., Danwar, H.S., (2011), Algorithms for Sindhi Word Segemtnatin Using Lexicon Driven Approach, International Journal of Academic Research, Vol. 3. No.3. May, 2011.

[22] Ismaili, I.A., Bhatti, Z., Shah, A. A., “Development of Unicode based bilingual Sindhi-English Dictionary”. Mehran University Research Journal of Engineering & Technology Volume 31, No. 1, January 2012.

[23] Bhatti, Z., Ismaili, I.A., Shaikh, A. A., Soomro, W. J. “Spelling Error Trends and Patterns in Sindhi”. Journal of Emerging Trends in Computing and Information Sciences, Vol. 3, No.10, 2012.

[24] Bhatti, Z., Ismaili, I.A., Khan, W., Nizamani, A. S., “Development of Unicode based Sindhi Typing System”, Journal of Emerging Trends in Computing and Information Sciences, Vol. 4 No. 3, 2013.