World Journal of Engineering Research and Technology
WJERT
www.wjert.org
SJIF Impact Factor: 5.218

BANGLA ROOT WORD CORPUS

Kazi Wohiduzzaman*[1] and Sabir Ismail[2]

[1]Department of Electrical and Electronics Engineering, Metropolitan University.

[2]Department of Computer Science and Engineering, Shahjalal University of Science and Technology.

Article Received on 30/03/2018 Article Revised on 19/04/2018 Article Accepted on 09/05/2018

***Corresponding Author**
Kazi Wohiduzzaman
Department of Electrical and Electronics Engineering, Metropolitan University.

ABSTRACT

Bangla is a very rich language, two hundred and thirty million world populations speak in Bangla. Hence, the computerization of this language is the inevitable need today. Unfortunately, a very few research work have been done in this field due to resource scarcity. The effort of Bangla computerization could not reach up to satisfactory level compared to other languages. We are going to describe the efficient algorithm of finding Bangla root word. This root word corpus store valid root word with its inflectional forms. We used Bangla word resource from Bangla Newspaper, Blogs etc. In our proposed Algorithm firstly collected 200000 words, filtered some incorrect words form and removed duplicate words from the list. Finally, stored 60000 unique Bangla word in a word list. This paper represents an efficient technique of find out real root of a word. In natural language processing, it's very important to find the real root of a word for information retrieval, document categorization etc.

KEYWORDS: *NLP, Bangla, Suffix, Root Word, Stemming.*

1. INTRODUCTION

Root word corpus is the guide to use appropriate words to mention the proper situations. In linguistic, a root word holds the most basic meaning of any word. A root word has no suffix or prefix, it's the heart of word. There the words are explained by their meanings synonyms and using process. So that man can learn the proper way to use words in his speaking, writing using a word corpus. Bangla is an ancient language. This language is a highly inflected language. There are many different ways to create a Bangla word. Normally we know that

root word contain most significant part of a word. One root word has many inflection forms. So, we try to build up a corpus for Bangla root word with its inflectional forms.

For example:

"বলেছিলাম" is a verb inflectional form of root "বল".

বলেছিলাম=বল+ েছিলাম । Here, "েছিলাম" is a suffix.

If we create a word corpus for valid root word in Bangla language then this dictionary help to spelling checking, parts-of-speech tagging, word stemming, word clustering, word sense disambiguation. So, a valid root word corpus is very useful and helpful to us. Check correct spelling of a word.

Example: "দেখ, দেক" । Here "দেক" is not a correct spelling.

POS tagging is the process of marking up a word in a text as corresponding to a particular part of speech, based on both its definition, as well as its context. Word stemming is the process for reducing inflected (or sometimes derived) words to their stem, base or root form—generally a written word form.

Example: ছেলেটা = ছেলে + টা ।

(Here, "ছেলে" is a root and "টা" is a suffix).

Word clustering is a process of grouping the word, which is semantically similar and can thus bear a specific meaning.

Example: "ছেলে, মেয়ে, ছাত্র" those words have maximum similar inflectional forms.Word sense disambiguation is an open problem of natural language processing, which governs the process of identifying which sense of a word is used in a sentence, when the word has multiple meanings.

Example: আমি চলে যান ।

Normally "যান" in use of honor person but here subject is first person singular number.

In this paper, we represent an efficient method of finding root word corpus. We depict three different technique to find out root word for any word of Bangla language.

LITERATURE REVIEW

There is a very few work we find about suffix tree related, there is a paper describes a method for the development of Bangla In conversion within the framework of the Universal Networking Language (UNL). In this paper, a pioneer work proposes that analyzes the Bangla words morphologically from which we obtain Roots and Primary suffixes (Kit Pretty) and develops some rules for Bangla Root, and Primary Suffix for the UNL. Describes grammatical attributes for Bangla root and Primary suffixes and use of morphological analysis (M.N.Y. Ali et and ms, 2010). Another research are presenting a pioneer work that aims to contribute to Development of Models for Bangla Dictionary Entries and Analysis of Grammatical Attributes of Bangla words such as Bangla Roots, Krit Prottoy and Kria Bivokti which will help to create a doorway for converting the Bangla Sentence to UNL and vice versa and subside the barrier between Bangla to other languages. Paper presents Models for Bangla Root, Krit Prottoy and Kria Bivokti which avail dictionary entries of the root, Krit Prottoy and Kria Bivokti (Z. Hossain and ms, 2012).

For the development of Bangla Enconversion within the framework of the Universal Networking Language (UNL). They also discuss some issues and problems related to the UNL representation that affects the quality of generation. Additionally, the Iingware engineering is introduced as a technique to enhance the quality and increase the development efficiency (UzZaman and ms, 2006). This paper proposes a pioneer work that analyzes the Bangla words morphologically from which we obtain Roots and Krit Prottoy (Primary suffixes) and develops some rules for Bangla Root, Krya Bivokti (Verbal suffixes) and Primary Suffix for the UNL. The goal is to include Bangla in this system with less effort (M.F. Mridha and ms, 2010). A new algorithm, called Semantic Suffix Tree Clustering (SSTC), to cluster web search results containing semantic similarities. The distinctive methodology of the SSTC algorithm is that it simultaneously constructs the semantic suffix tree through an in-depth and on-breadth pass by using semantic similarity and string matching. The semantic similarity is derived from the Word Net lexical database for the English language. SSTC uses only subject-verb-object classification to generate clusters and readable labels. The algorithm also implements directed pruning to reduce the sub-tree sizes and to separate semantic clusters. Experimental results show that the proposed algorithm has

better performance than conventional Suffix Tree Clustering (STC) (Alam and ms, 2010). SSTC can cluster documents that share a semantic similarity. The specific cluster is returned in a readable form. Additionally, the SSTC can improve the performance of approaches that use the original STC algorithm because it can cluster semantically similar documents, reduce the number of nodes and reach higher precision (J. Janruang and ms, 2011).

The rich morphology of Tamil enables the Enconversion process to be based on morpho-semantic features of the words and their preceding and succeeding context. The use of case relation indicating morphological suffixes, POS tag and word level semantics allows the rule based Enconversion to be independent of the syntactic structure of the sentence. These UNL graphs are used to build a conceptual level index. An Interlingua, language independent framework that encapsulates the meaning of sentences in terms of concepts and relations has been the focus of researchers working in areas that deal with multiple languages like machine translation and cross-lingual information retrieval. One such Interlingua framework called KANT, which has been targeted towards technical text in controlled sub-domain. UNITRANS, a bidirectional system, operates cross-linguistically but still accounting for knowledge that is specific to each language (J Balaji and ms, 2011). An on-line algorithm is presented for constructing the suffix tree for a given string in time linear in the length of the string. The new algorithm has the desirable property of processing the string symbol by symbol from left to right. It always has the suffix tree for the scanned part of the string ready. The method is developed as a linear-time version of a very simple algorithm for (quadratic size) suffix trees. Regardless of its quadratic worst case, this latter algorithm can be a good practical method when the string is not too long. Another variation of this method is shown to give, in a natural way, the well-known algorithms for constructing suffix automata (DAWGs) (E Ukkonen, thesis) (Majumder and ms, 2006).

PROPOSED METHOD

In our proposed Algorithm at first, collected 200000 words from different resources, then removed some words also which are not in correct form and duplicate also. Finally, we stored 60000 unique Bangla word in the word list. We constructed a suffix list with the help of many Bangla grammar books. In suffix list has 540 suffixes. Among all suffix, we stored 100 suffixes from different Bangla grammar books. We have written an algorithm for creating the rest list of the suffix. We selected 50 words from the word list, where some words are Nouns, some are Verbs and some are Adjectives for creating a short list. In short list, all words are

well-known root word. Then we try to compare each word in a short list with every other word in the word list. If a word in the word list starts with a word in the short list then we chopping up substring between two words. And chopped word stored as a valid suffix in suffix list.

A. Method One

In this technique one word list is define a vector which contains all words, and also another vector suffix list contains all suffix. And store word is a HashMap. In the store, word will store root word as a key and an array list using to store the variation of root word as value. Each word from word list temporarily named word, if the length of the word is greater than two then this word compare with every other word from the word list. Second time every each word from word list temporarily called tamp word. If tamp word starts with a word then we chop up substring between tamp word and word. In next step, we temporarily stored chopped word as tamp suffix. We search in suffix list if tamp suffix is a valid suffix at that time we stored tamp word as a value under the key word in store word. If tamp suffix is not a valid suffix finally we stored word as a key and value are null in store word.

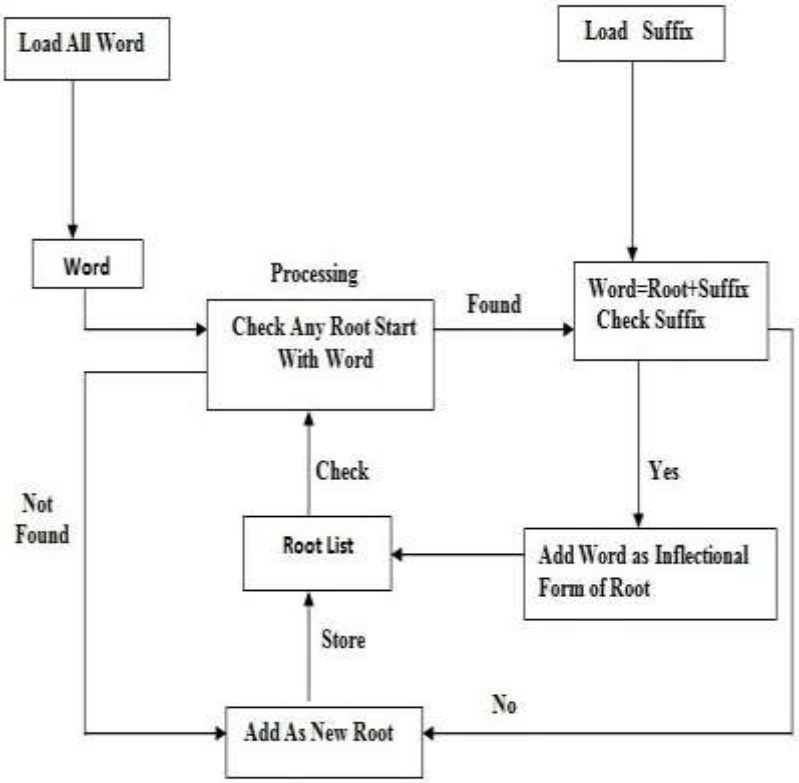

Figure 1: Flow Chart of method One Algorithm.

B. Method Two

In method two-word list is a vector which contains all words, another vector suffix list contains all suffix. And store word is a Hash Map. In the store, word will store root word as a key and an Array list using to store the variation of root word as value. Each word from word list temporarily we named word. If the length of the word is greater than two then this word compare with every other word from the word list. Second time every each word from word list temporarily we called tamp word. If tamp word starts with a word then we chop up substring between tamping word and word. In next step we temporarily stored chopped word as tamp suffix. We search in suffix list if tamp suffix is a valid suffix at that time we stored tamp word as a value under key word in store word. If tamp suffix is not a valid suffix finally we stored word as a key and value are null in store word. In this method, if the word has two or more than two valid variations in word list we stored that word as a root word in store word.

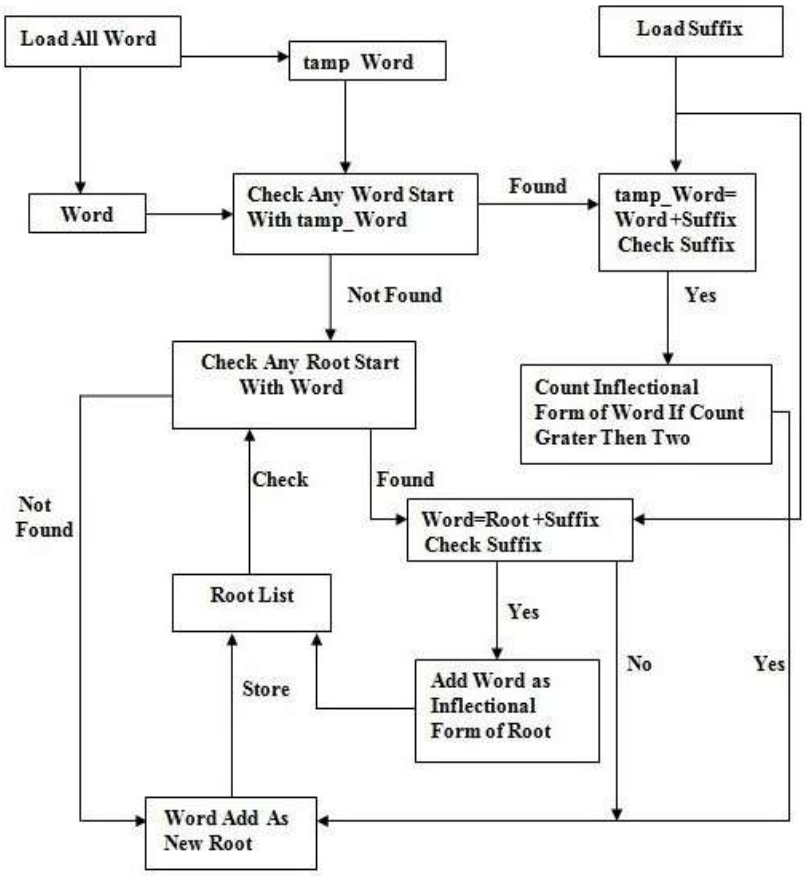

Figure 1: Flow Chart of method Two Algorithm.

C. Method Three

In the method, the three-word list is a vector which contains all root words. All words in word list are we stored from method two, another vector suffix list contains all suffix. And store root word is a vector. In the store, root word will store all new root words. Each word from word list temporarily we named word. This word compares with every other word from the word list. Second time every each word from word list temporarily we called tamp word. We temporarily stored maximum similar string between word and tamp word in new root. Then we chop up substring between tamping word and word. In next step, we temporarily stored chopped word as tamp suffix. We search in suffix list if tamp suffix is a valid suffix at that time we stored new root as a new root word in store root word. If tamp suffix is not a valid suffix finally we stored word as a root word in store root word.

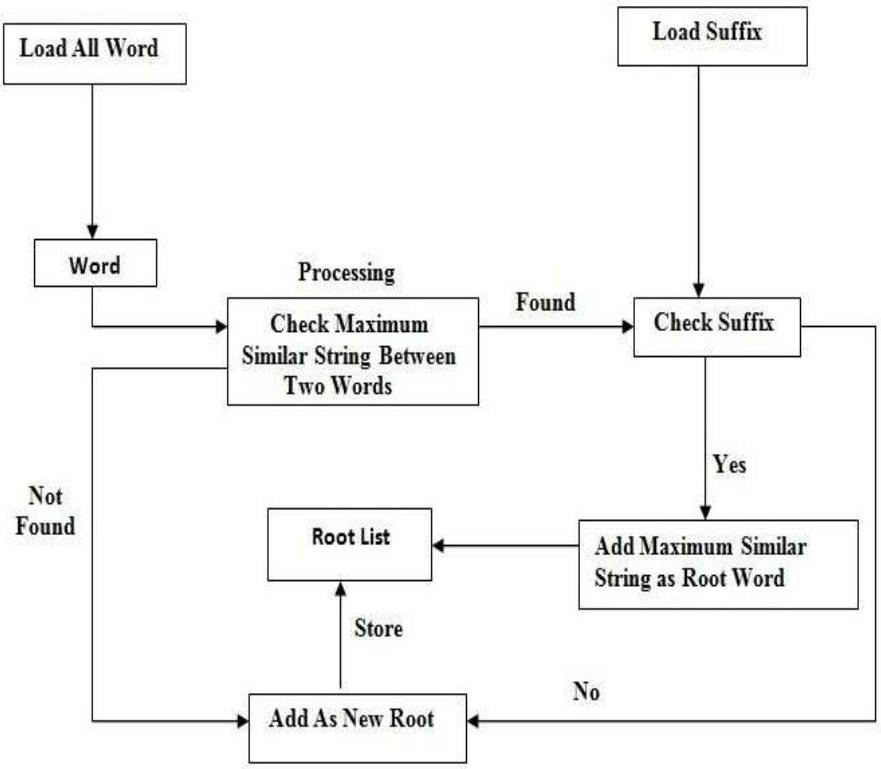

Figure 1: Flow Chart of method Three Algorithm.

RESULT ANALYSIS

D. The result of Method One

We worked with 582 words and our proposed algorithm find 67 words as root words. Almost 70% words have inflection form. We fail to tag some root word correctly because of inflectional form of that word not available in our corpus. And also some suffixes are valid

for some words at the same time those suffixes are not a correct inflectional form of some words.

For example, "অক্ষ" is a valid root word and "র" is a valid suffix but "অক্ষর" is a valid root word. In our algorithm "অক্ষর" stored as inflectional form of root word "অক্ষ". We tring to Overcome this type of problem in method two. Output accuracy of this method is 60%.

E. The result of Method Two:

We work with 582 words and our proposed algorithm find 84 words as root words. Almost 70% words have inflection form. We fail to tag some root word correctly because of inflectional form of that word not available in our corpus. And also some suffixes are valid for some words at the same time those suffixes are not a correct inflectional form of some words. In this method, we are tried to solve the problem which created in method one.

For example, in this algorithm "অক্ষ" and "অক্ষর" both are stored as valid root word. Because "অক্ষর" and "অক্ষ" both words have more than two inflectional form in word list. At the same time some new problem has been created . "অংশ"and "অংশগুলি" both are stored as valid root word. But "অংশগুলি" is an inflectional form of "অংশ". We tryed to slove this type of problem in method three. Output accuracy of this method is 46.6%.

F. The result of Method Three

In this method, we work with stored root word from method two. In word list, some words are valid root words and some are an inflectional form of the root word. Our proposed algorithm find 21 words as root words for 84 input words. So, after analysis the output of this method we realized that if want to overcome this type of problem then at first stage we mast focused the term is Word Sense Disambiguation. In Word Sense Disambiguation we can find out a correct sense of a word. Which type of variations performs for a valid root word? Output accuracy of this method is 71.43%.

CONCLUSION

Bangla language has a wide range of grammatical rules including parts of speech, number, gender, person, mood, affixes. They are used in Bangla language in a way than the English language. There are several methods of building Bangla word. In this research, we have tried to propose an efficient way to made bangle root word dictionary. Our main effort was to

create a valid suffix list and analysis the suffix list. For making valid root word storage we have created three types of methods. The output accuracy of our created algorithm is relying on suffix list and input word list.

BIBLIOGRAPHY

G. Web Links

1. http://www.prothom-alo.com/ (সর্বাধিক প্রচারিত বাংলা দৈনিক পত্রিকা. Most popular bangla daily newspaper).
2. http://www.kalerkantho.com/ *(*Strongest panel of journalist in Bangladesh put their hands together to reveal the best out of the facts for the nation Kaler kantho).
3. http://www.amardesh.com/ (*Amar-desh* (Daily *Amar Desh*) Bangladesh Newspaper Provides Bangladesh and International news, business, science, technology, sports, movies news).

H. Core Reading

1. "উচ্চতর বাংলা ভাষারীতি" –প্রফেসর ড. আলাউদ্দিন আল আজাদ , ড. মনন অধিকারী , রুহুল আমিন বাবুল।
2. " বাংলা ব্যাকরণ" –নবম-দশম শ্রেণি ।
3. "বাংলা ব্যাকরণ" – ড. সুনীতিকুমার চট্টোপাধ্যায় ।
4. "বাংলা পদগুচ্ছের সংগঠন" - উদয়কুমার চক্রবর্তী ।

REFERENCES

1. Alam, Firoj, S. M. Habib, Dil Afroza Sultana, and Mumit Khan. "Development of annotated Bangla speech corpora.", 2010.
2. Esko Ukkonen, 'on-line construction of suffix trees', Department of Computer Science, University of Helsinki, A thesis paper.
3. Jongkol Janruang, and Sumanta Guha, 'Semantic Suffix Tree Clustering', First IRAST International Conference on Data Engineering and Internet Technology (DEIT), 2011.
4. J Balaji, T V Geetha, Ranjani Parthasarathi, Madhan Karky (2011), 'Morpho-Semantic Features for Rule-based Tamil Enconversion', International Journal of Computer Applications (0975 – 8887), July 2011; 26(6).

5. Md. Nawab Yousuf Ali, Md. Zakir Hossain, Shahid Al Noor, Jugal Krishna Das., 'Development of Analysis Rules for Bangla Root and Primary Suffix for Universal Networking Language' 'International Conference on Asian Language Processing', 2010.
6. Md. F. Mridha, Md. N. Huda, Chy M. Rahman, JK Das, 'Development of Morphological Rules for Bangla Root, Verbal Suffix and Primary Suffix for Universal Networking Language.', 6th International Conference on Electrical and Computer Engineering ICECE 2010, 18-20 December 2010, Dhaka, Bangladesh, 2010.
7. Majumder, Khair Md, and Yasir Arafat. "Analysis of and observations from a Bangla News Corpus.", 2006.
8. UzZaman, Naushad, and Mumit Khan. A comprehensive Bangla spelling checker. Center for research on Bangla language processing (CRBLP), BRAC University, 2006.
9. Zakir Hossain, Shahid Al Noor, Muhammad Firoz Mridha, 'Some Proposed Standard Models for Bangla Dictionary Entries of Bangla Morphemes for Universal Networking Language.', IJCSNS International Journal of Computer Science and Network Security, November 2012; 12(11).

APPENDIX

Sample Input

অংশ	অংশও	অংশগুলির	অংশগুলোও
অংশই	অংশকে	অংশগুলো	অংশগুলোতে
অংশটি	অংশীদার	অংশগ্রহণে	অংশগ্রহণরত
অংশটা	অংশবিশেষ	অংশজুড়ে	অংশগ্রহকারীদের
অংশেও	অংশে	অংশটিতে	অংশগ্রহণকারী
অংশটুকু	অংশটির	অংশগ্রহণ	অংশগ্রহণের

Sample Output One

Root	**Inflectional Form Of Root**
অংশ:	অংশ, অংশই, অংশও, অংশকে, অংশগুলির, অংশগুলো, অংশগুলোও, অংশগুলোতে, অংশটা, অংশটি, অংশটিতে, অংশটির, অংশটুকু, অংশটুকুই, অংশটুকুও, অংশসহ, অংশে, অংশেই, অংশেও, অংশের, অংশেরই
অংশগ্রহণ:	অংশগ্রহণ, অংশগ্রহণও, অংশগ্রহণকে, অংশগ্রহণে, অংশগ্রহণের

Sample Output Two

Root	**Inflectional Form Of Root**
অংশ:	অংশ, অংশই, অংশও, অংশকে, অংশগুলির, অংশটা, অংশসহ
অংশগুলো:	অংশগুলো, অংশগুলোও, অংশগুলোতে
অংশগ্রহণ:	অংশগ্রহণ, অংশগ্রহণও, অংশগ্রহণকে, অংশগ্রহণে, অংশগ্রহণের
অংশটি:	অংশটি, অংশটিতে, অংশটির
অংশে:	অংশে, অংশেই, অংশেও, অংশের, অংশেরই

Sample Output Three

Input	Output
অংশ অংশে অংশটি অংশীদারি অংশীদারিত্ব	অংশ অংশীদার অংশীদারি