Lemmatization Algorithm Development for Bangla Natural Language Processing

Md. Kowsher
Dept. of Applied Mathematics
Noakhali Science and Technology University
Noakhali, Bangladesh.
ga.kowsher@gmail.com

Anik Tahabilder
School of Engineering + Technology
Western Carolina University
Cullowhee, NC-28723, USA
tahabilderanik@gmail.com

Md Murad Hossain Sarker
Dept. of Information and Communication Technology
Comilla University
Comilla, Bangladesh
mh6367828@gmail.com

Md. Zahidul Islam Sanjid
Dept. of Computer Science and Engineering
BRAC University
Dhaka, Bangladesh.
sanjidmzi53@gmail.com

Nusrat Jahan Prottasha
Dept. of Computer Science and Engineering
Daffodil International University
Dhaka, Bangladesh.
nuaratjahan1234561234@gmail.com

***Abstract*— Natural language processing (NLP) finds enormous applications in autonomous communication, while lemmatization is an essential preprocessing technique for simplification of a word to its origin-word in NLP. However, there is scarcity of effective algorithms in Bangla NLP. This leads us to develop a useful Bangla language lemmatization tool. Usually, some rule base stemming processes play the vital role of lemmatization in Bangla language processing as there is lack of Bangla lemmatization tool. In this paper, we propose a Bangla lemmatization framework using three effective lemmatization techniques based on data structures and dynamic programming. We have used Trie algorithm and developed a mapping algorithm named "Dictionary Based Search by Removing Affix (DBSRA)" based on data structure. We have applied both Trie and DBSRA lemmatization and selected the better one by considering the Levenshtein distance between the lemma and the original word. Eventually, we have experimented with Bangla language lemmatization among all three techniques and the framework. Among the three proposed techniques, the DBSRA performed better compared to others with an accuracy of 93.1 percent. The framework, developed by fusing three algorithms, came out with the highest efficiency of 95.89 percent.**

***Contribution*—This paper presents the development of three lemmatization algorithms and their fusion to develop a framework for Bangla Natural Language Processing.**

Keywords*— *Bangla NLP, lemmatization, Trie, DBSRA , corpus.

I. INTRODUCTION

Nowadays, we live in an era of automation with the help of computers. For automation, it is essential to develop computer programs to process and analyze large amounts of natural language data. However, this natural language processing (NLP) faces huge challenges as it involves a good understanding of natural languages [1]. Bangla is one of the complicated languages. But there is no effective lemmatization technique in this language processing. A Lot of affixes (suffixes and prefixed) are there in a word in this natural language [2]. Lemmatization is a simple process to extract the root-word for natural language understanding [3]. Lemmatization has been used in a variety of real-world applications, such as text mining, chatbot, questions and answering, and so one [4]. Usually, lemmatization is done by dynamic programming based Levenshtein distance and data structure based Trie (keyword tree) algorithms [5].

In this paper, we have attempted to develop an effective lemmatization framework for the Bangla natural language processing and developed three data structure and mapping based Bangla lemmatization algorithms. At first, we used the Trie algorithm based on prefixes. After that, we proposed a mapping based new algorithm titled Dictionary-Based Search by Removing Affix (DBSRA). We have done experimentations using the Bangla text corpus to compare the accuracy and complexity of the approaches. Then we combined the Levenshtein distance-based approach, Trie algorithm and DBSRA algorithm for developing our new framework named Bangla Lemmatization Framework. The work of this paper can be summarized as below:

- We have presented three Bangla lemmatization techniques including Dictionary Based Search by Removing Affix (DBSRA), a new technique of lemmatization for Bangla language.
- We have introduced a Bangla lemmatization framework to overcome the limitations of existing techniques by implementing three methods based on data structure and dynamic programming.
- We have performed experiments and showed complexity analysis along with show comparisons among proposed algorithms with the framework.

In the following section II, we have discussed some relevant works that have been done in this field. In section III, we have described our proposed method. In section IV, we

978-1-7281-9331-1/20/$31.00 ©2020 IEEE

have described the experiment as well as the results, and we have finished with the conclusion in section V.

II. Related Work

D. Namly et al. manifested a new lemmatizer that conjoins a lexicon-based methodology with a machine-learning-based procedure for the Arabic language [6]. In 2020 R. Stankovic et al. developed a machine learning and neural network model for Lemmatization and Morphosyntactic Tagging in Serbian language [7]. A. Chakrabarty et al. generated a BLSTM-CNN lemmatizer for Hindi, Marathi, French and Spanish languages incorporating limited context [8]. A mixed approach was adopted by M. Boudchiche et al. for Arabic text lemmatization utilizing statistical procedure based on the hidden Markov models [9]. A hidden Markov model for Arabic Root Extraction was introduced by A. Boudlal et al. in 2011, where the words and the possible roots represented the hidden states of the model [10]. In 2008 E. Al-Shammari et al. manifested an effective lemmatization algorithm for Arabic text and claimed that lemmatization outperforms stemming in terms of mining Arabic text [11]. A deep encoder-decoder architecture for the lemmatization of multiword expressions was proposed by M. Schmitt et al. and was evaluated for Italian, French, Polish and Portuguese language [12]. In 2012, a root-based non-statistical Arabic lemmatizer algorithm was designed by T. El-Shishtawy with the help of different Arabic language knowledge resources [13]. T. Bergmanis et al. showed a lemmatizer titled "Lematus", a model to improve performance on inconspicuous and ambiguous words. Moreover, they evaluated its lemmatization accuracy across 20 dialects [14]. T. Muller et al. introduced a modular log-linear model named "Lemming" for lemmatization and tagging and claimed its effectiveness in six languages [15]. T. Korenius et al. investigated the performance of four hierarchical clustering methods and hypothesized that lemmatization with split compounds provides a better clustering result than stemming in Finish language in 2004[16]. A. Gesmundo et al. described [17] a new method of lemmatization by formalizing lemmatization as a form tagging task and evaluated this scheme on eight different languages. P.

III. Proposed Work

In this work, we have presented the development of lemmatization for Bangla Language processing and understanding. We have used Trie, Levenshtein distance and DBSRA algorithms to find the lemma of a Bangla word. But each of this algorithm has some limitations. So, we combined all the algorithms and made a framework. Before applying the lemmatization algorithms, we prepared data in three steps, such that collecting data sets, preprocessing of these, and running the preprocessed data with proposed algorithms. The proposed work was completed by a few steps, including data collection, preprocessing, and algorithm design, as shown in figure 1.

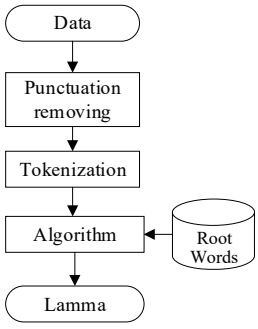

Fig.1. Flow chart of proposed work

In the lemmatization of the Bangla NLP, mainly two kinds of the corpus are extracted. The first one is the Bangla root word corpus for finding root words from the inputted texts. We used 28,324 Bangla root words. The other one is the Bangla punctuations corpus. We stored all kinds of punctuations or special characters of the Bangla language.

To test the performance of the proposed algorithm, we collected 376 different news from the popular Bangla newspaper 'The Daily Prohtom Alo' as test data.

A. Data Preprocessing

We normalized the inserted file of texts by preprocessing that includes punctuation removing, and tokenizing.

Let's consider the example below:

Input: 'আমাদের দেশ নদীমাতৃক দেশ। এই দেশের প্রধান নদী গুলো হল পদ্মা, মেঘনা, যমুনা।'

Since punctuation mark doesn't need lemmatization, we have removed the punctuation marks at the beginning.

output: 'আমাদের দেশ নদীমাতৃক দেশ এই দেশের প্রধান নদী গুলো হল পদ্মা মেঘনা যমুনা'

Tokenization is an NLP technique that breaks or splits a stream of texts into words, sentences, symbols, phrases or other elements that are meaningful. After removing punctuation mark, we did Tokenization of the input sentence. The input and output of the tokenization is as shown below:

Input: আমাদের দেশ নদীমাতৃক দেশ এই দেশের প্রধান নদী গুলো হওয়া পদ্মা মেঘনা যমুনা

output: আমাদের; দেশ; নদীমাতৃক; দেশ; এই; দেশের; প্রধান; নদী; গুলো; হওয়া; পদ্মা; মেঘনা; যমুনা;

B. Lemmatization Techniques for Bangla Language

In the present work, three kinds of Bangla lemmatization techniques based on data structure and dynamic programming are imparted as BNLP. The first one is the string matching and dynamic programming algorithm called Levenshtein Distance. The second technique is 'Trie' based on a popular data structure tree. It is applied in the data storing process and searching with the lowest space, time complexity. The third algorithm is Dictionary Based Search by Removing Affix shortly 'DBSRA', that is originated as a new algorithm of Bangla lemmatization based on dictionary.

C. Levenshtein Distance

In computational linguistics, the Levenshtein Distance is a procedure using dynamic programming to measure the smallest figure of a single character that is enunciated to edit or change between two words (i.e. Insertions, Deletions or Substitutions). In the applications of NLP, it is widely applied for the correct spelling of words.

$$lev_{a,b}(i,j) = \begin{cases} max(i,j) & if\ min(i,j) = 0 \\ min \begin{cases} lev_{a,b}(i-1,j) + 1 \\ lev_{a,b}(i,j-1) + 1 \\ lev_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} \end{cases}$$

In order to find the lemma, we have used the following formula.

$$Lemma(word) = min\ (edit\ (word,\ root\ words_{\ i})$$
$$Where\ i = one\ to\ the\ number\ of\ the\ root\ word$$

In this paper, we have used the Levenshtein Distance for the sake of Bangla language lemmatization. Here is an instance of how Levenshtein Distance's technique works for Bangla lemmatization.

Let's take two Bengali words 'অধিক' (adhek, more) and 'অধিকার' (adhekar, right). If an inputted word is 'অধিকারী' (adhikari, owner) which actually comes from 'অধিকার' (adhekar, right). The edit distances of 'অধিকারী' (adhikari, owner) with respect two corpus words 'অধিক' (odhek, much) and 'অধিকার' (odhekar, right) are respectively 3 and 1 where minimum edit distance is 1. So the root word of 'অধিকারী' (adhikari, owner) is 'অধিকার' (adhekar, right). The edit calculation between the word pair অধিক-অধিকারী and the between the word pair অধিকার-অধিকারী are shown in table 1 and table 2.

TABLE I. THE EDIT OF 'অধিক-অধিকারী'

		অ	ধ	ি	ক
	0	1	2	3	4
অ	1	0	1	2	3
ধ	2	1	0	1	2
ি	3	2	1	0	1
ক	4	3	2	1	0
া	5	4	3	2	1
র	6	5	4	3	2
ী	7	6	5	4	3

TABLE II. THE EDIT OF 'অধিকার-অধিকারী'

		অ	ধ	ি	ক	া	র
	0	1	2	3	4	5	6
অ	1	0	1	2	3	4	5
ধ	2	1	0	1	2	3	4
ি	3	2	1	0	1	2	3
ক	4	3	2	1	0	1	2
া	5	4	3	2	1	0	1
র	6	5	4	3	2	1	0
ী	7	6	5	4	3	2	1

D. Trie

Trie or Prefix Tree is a tree-based data structure and exoteric for the mechanisms to store data. In our system, we narrated a Trie algorithm to implement in NLP approaches for Bangla word lemmatization. The mathematical expression of the formula is as follows:

$$Lemma(word) = min\,(distance\,(word_j, root\ words_i)$$
$$Where\ i = one\ to\ the\ number\ of the\ root\ word$$
$$j = one\ to\ the\ length\ of\ the\ highest\ prefix$$

Let's consider, there are four words in a Trie such as 'অতীত' (otet, past), 'অতিথি' (otethi, guest), 'লোভ' (lov, greed) and 'লোক' (lok, people). The common prefix of the words 'অতীত' (otet, past) and 'অতিথি' (otethi, guest) is 'অত' (oto) and the common prefix of the words 'লোভ' (lov, greed) and 'লোক' (lok, people) is 'লো' (lo). From the stored data, we have to check if a new word is stored in the tree or not. If all characters of the word are found in the tree then the word is counted as lemma word. Otherwise, we need to find the closest end node. The flow chart of using trie for lemmatization is shown in figure 2.

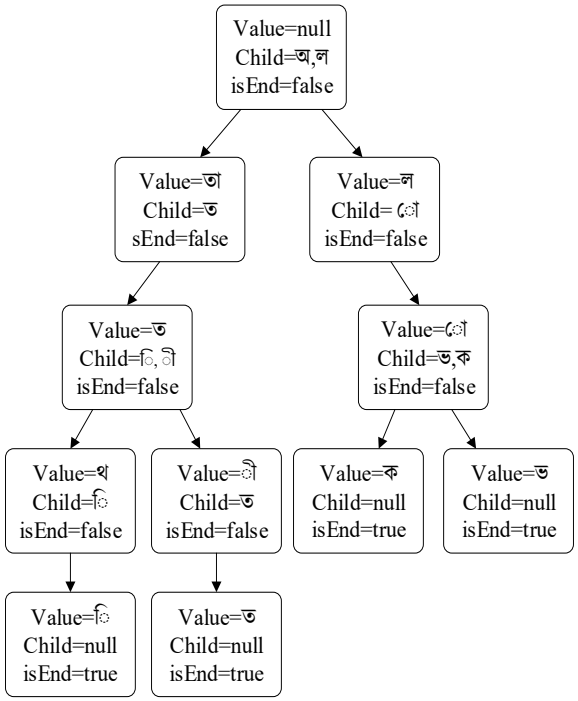

Fig.2. Data Storing in Trie

There are lots of words in Bangla that start with a prefix. So, the Tire technique goes to wrong. For example, for an inputted word 'অতিলোভ' (otelov, too greed), searching 'অতিলোভ' (otelov, too greedy) in previous Trie, we get 'অতিত' (otet, past) instead of 'লোভ' (lov, greed). So Trie does not work properly for such a case of the prefix. In this case, by carrying away the first character then the word is 'তিলোভ' (telov) and search in a tire and note the distance from the ended or invalid position to the next root. Identically, we gain the word as 'লোভ (lov, greed), search in Trie and note the distance from the ended or invalid position to the next root, which is zero. So, our desired root word is 'লোভ' (lov, greed). The solution of the problem by using trie search is shown in a flowchart shown in figure 3.

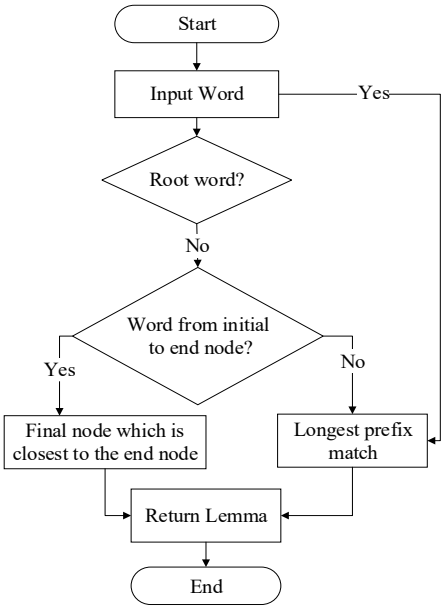

Fig.3. Search in the Trie

E. Dictionary-Based Search by Removing Affix (DBSRA)

We have fabricated a new lemmatization mapping algorithm named as 'Dictionary Based Search by Removing Affix (DBSRA). In this technique, at first dismissed the i-th character from the desired word (which we want to transfer into root word) where i = 0, 1, 2, ..., (length of word -1) and remove last j-th (where j = length of word n, n-1, n-2, ..., 1) character then this algorithm investigates in our corpus if the intentional words are in the root words corpus or not. But when the root word corpus is in massive volume, the searching will not be performed well, and time complexity will be high. So, the whole corpus has been mapped by the assist of the dictionary-based program, and the words can be investigated within the lowest time complexity. Let's 'অধিকারী' (adhekari, owner) is our desired word. We can make up the word by removing affix, as shown in table 3.

TABLE III. ARRANGEMENT OF BANGLA WORD

অধিকারী	ধিকারী	িকারী	কারী	ারী	রী	ী
অধিকার	ধিকার	িকার	কার	ার	র	
অধিকা	ধিকা	িকা	কা	া		
অধিক	ধিক	িক	ক			
অধি	ধি	ি				
অধ	ধ					
অ						

In the abobe table, three words (underlined in the table) 'অধিকার' (adhekar, right), 'অধিক' (adhek, more) and 'কার' (car) found in the mapping function. But the most largest word is 'অধিকার' (adhekar, right) so the desired root word of 'অধিকারী' (adhekari, owner) is 'অধিকার' (adhekar, right. If the DBSRA does not find out any mapping word, then the word is counted as an unknown word.

$$Lemma(word) = max\ (mapping\ (word_{j,k}\ , root\ words_i)$$

$$j = removing\ character\ from\ the\ backend\ of\ word$$

$$k = removing\ character\ from\ the\ front\ of\ j-th\ word$$

F. Bangla Lemmatization framework

It is mentionable that the proposed three algorithms are not sufficient for 100% lemmatization of Bangla language. The major problems are summarized as

- Trie is not the right choice for unknown words
- DBSRA does not work when lemma words remain with other Bangla characters, which is not in inserted words.
- Levenshtein Distance does not perform for unknown words, but the primary issue is high execution time.

In order to recover these problems and get the best accuracy with the lowest execution time and space, we tied three algorithms with a bond as the Bangla lemmatization framework. First, we need to run every word with both Trie and DBSRA. Because there are lots of words those work in DBSRA but not in Trie. The whole process can be speared with four views.

The contribution of the Bangla Lemmatization Framework is summarized below:

- If the lemma of a word has no difference between Trie and DBSRA then it is counted as the root word.
- Otherwise, they are checked with the target word by Levenshtein Distance. The word with the minimum edit will be the root word.
- If the obtained probability of edit of both words is greater than 50%, then the target word is counted as an unknown word.
- The unknown words are processed by removing the longest suffix.

The Bangla lemmatization framework is shown below in figure 4.

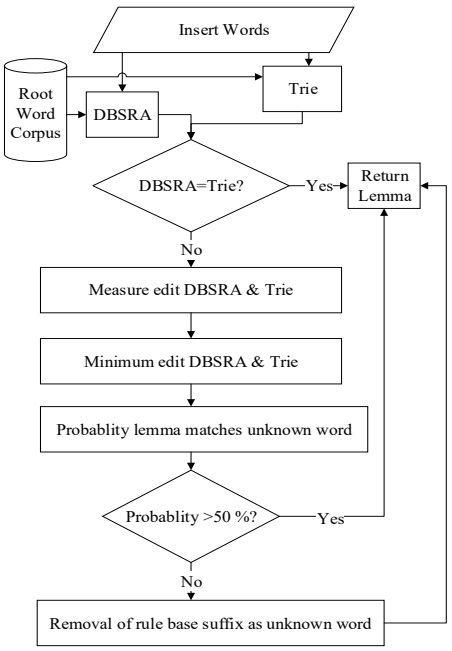

Fig.4. Bangla Lemmatization Framework

IV. Experimental Results

In this work, we have used the Bangla lemmatization framework to convert the words into its origin. To install the 'Bangla lemmatization framework', we have run the following command in the command window.

```
pip install BnLemma
```

In order to use the framework, we have imported it as lm. We gave a string input as "মানুষের জীবনটা পাচ্ছেন তাই কাজে লাগানো দরকার আমাদেরকে" and got proper output string of "মানুষ জীবন পাওয়া তাই কাজ লাগা দরকার আমাদের" as shown in figure 5.

```
>>> from BnLemma import lemmatization as lm
>>> s = "মানুষের জীবনটা পাচ্ছেন তাই কাজে লাগানো দরকার আমাদেরকে"
>>> s = lm.lemma(s)
>>> print(s)
Output: "মানুষ জীবন পাওয়া তাই কাজ লাগা দরকার আমাদের"
```

Fig.5. Using the Bangla lemmatization framework

A. Testing data

For testing the algorithms as well as the framework, 67 articles have been collected from Bangla popular newspaper 'The Daily Prothom Alo' covering the recent news of Bangladesh, including news about politics, entertainment, sports, educations, science and technology, etc. The 67 articles contain white-space and punctuation separated 17,848 words. The number of words is reduced to 16132 after cleaning and removing stop words.

B. Model Performance

The performances of the three methods and our developed Bangla Lemmatization Framework are evaluated based on accuracy, time and space complexity.

To be computerized the Bangla words and information matching, lemmatization plays a crucial role. There are very few strategies for Bangla lemmatization. Table IV shows the time and space complexity performance of the Bangla lemmatization.

TABLE IV. Time and Space Complexity for Single Data

Name	Levenshtein Distance	Trie	DBSRA	Bangla Lemmatization Framework
Time Complexity	O(N*T*A)	O(T)	O(T)	O(T)
Space Complexity	O(T*A)	O(L*S)	O(N)	O(N)

Where N = Size of root word's dictionary
T = Target word to find out the lemma
A = Average length of a dictionary word
L = Total number of nodes
S = Size of the alphabet (points to other tries)

The time and space complexity relation by increasing order is,

Time Complexity: O(T) < O(N*T*A)

Space Complexity: O(T*A) < O(L*S) < O(N)

The time complexity of Levenshtein Distance is O(N*T*A). When the corpus is too large, the result is not satisfactory because of slow performance. In Trie, time and space complexity are providing a negotiable result. The execution time complexity of DBSRA is O(T), which is always constant. Since the DBSRA maps every word of the corpus, it takes O(N) space complexity. On the other hand, the Bangla Lemmatization Framework shows the optimal time and space complexity which are O(T) and O(N).

C. Experimental result

We tested our model performance based on collecting real datasets. One of the segments among the datasets is shown so that the experimental result and difference among the proposed techniques can be analyzed.

Let's consider the segment is

ডেঙ্গুতে মৃত্যুর ১৬৯টি ঘটনার খবর পেয়েছে স্বাস্থ্য অধিদপ্তর। এর মধ্যে ৮০টি মৃত্যুর ঘটনা পর্যালোচনা করেছে রোগতত্ত্ব, রোগ নিয়ন্ত্রণ ও গবেষণা প্রতিষ্ঠান (আইইডিসিআর)। পর্যালোচনার পর ৪৭টি মৃত্যু ডেঙ্গুতে হয়েছে বলে নিশ্চিত করেছে পর্যালোচনা কমিটি। আজ রোববার বিকেলে স্বাস্থ্য অধিদপ্তরে আয়োজিত সংবাদ ব্রিফিংয়ে এ সব তথ্য জানান আইইডিসিআরের পরিচালক অধ্যাপক মীরজাদী সেব্রিনা।

After preprocessing the pre-processed segment is as below:
'ডেঙ্গুতে', 'মৃত্যুর', '১৬৯', 'ঘটনার', 'খবর', 'পাওয়া', 'স্বাস্থ্য', 'অধিদপ্তর', '৮০', 'মৃত্যুর', 'ঘটনা', 'পর্যালোচনা', 'করা', 'রোগতত্ত্ব', 'রোগ', 'নিয়ন্ত্রণ', 'গবেষণা', 'প্রতিষ্ঠান', 'আইইডিসিআর', 'পর্যালোচনার', '৪৭', 'মৃত্যু', 'ডেঙ্গুতে', 'হয়', 'বলা', 'নিশ্চিত', 'করা', 'পর্যালোচনা', 'কমিটি', 'আজ', 'রোববার', 'বিকেলে', 'স্বাস্থ্য', 'অধিদপ্তরে', 'আয়োজিত', 'সংবাদ', 'ব্রিফিংয়ে','তথ্য', 'জানানো', 'আইইডিসিআরের', 'পরিচালক', 'অধ্যাপক', 'মীরজাদী', 'সেব্রিনা',

The performance of Levenshtein Distance
['ডেঙ্গু', 'মৃত্যু', 'অংস', 'ঘটনা', 'খবর', 'পাওয়া', 'স্বাস্থ্য', 'দপ্তর', 'আখ', 'মৃত্যু', 'ঘটনা', 'পর্যালোচনা', 'করা', 'তত্ত্ব', 'রোগ', 'নিয়ন্ত্রণ', 'গবেষণা', 'প্রতিষ্ঠান', 'আইডিয়া', 'পর্যালোচনা', 'আখ', 'মৃত্যু', 'ডেঙ্গু', ' হয়', 'বলা', 'নিশ্চিত', 'করা', 'পর্যালোচনা', 'কমিটি', 'আজ', 'রোববার', 'বিকেল', 'স্বাস্থ্য', 'দপ্তর', 'আয়োজন', 'সংবাদ', 'ব্রিফিং', 'তথ্য', 'জানানো', 'আইডিয়া', 'পরিচালক', 'অধ্যাপক', 'ইরশাদ', 'জেব্রা']
We can see that, the algorithm outputs seven incorrect (underlined) words.

The performance of Trie
['ডেঙ্গু', 'মৃত্যু', '১৬৯', 'ঘটনা', 'খবর', 'পাওয়া', 'স্বাস্থ্য', 'দপ্তর', '৮০', 'মৃত্যু', 'ঘটনা', 'পর্যালোচনা', 'করা', 'রোগ', 'রোগ', 'নিয়ন্ত্রণ', 'গবেষণা', 'প্রতিষ্ঠান', 'সিনেমা', 'পর্যালোচনা', '৪৭', 'মৃত্যু', 'ডেঙ্গু', 'হয়', 'বলাকা', 'নিশ্চিত', 'করা', 'পর্যালোচনা', 'কমিটি', 'আজ', 'রোববার', 'বিকেল', 'স্বাস্থ্য', 'দপ্তর', 'আয়োজন', 'সংবাদ', 'ব্রিফিং', 'তথ্য', 'জানানো', 'রেখা', 'পরিচালক', 'অধ্যাপক', 'জাদু', 'ব্রিগেড']
Trie output five incorrect (underlined) words, which is less than the Levenshtein output.

The performance of DBSRA
['ডেঙ্গু', 'মৃত্যু', '১৬৯', 'ঘটনা', 'খবর', 'পাওয়া', 'স্বাস্থ্য', 'দপ্তর', '৮০', 'মৃত্যু', 'ঘটনা', 'পর্যালোচনা', 'করা', 'তত্ত্ব', 'রোগ', 'নিয়ন্ত্রণ', 'গবেষণা', 'প্রতিষ্ঠান', 'আইইডিসিআর', 'পর্যালোচন', '৪৭', 'মৃত্যু', 'ডেঙ্গু', 'হয়', 'বলা', 'নিশ্চিত', 'করা', 'পর্যালোচনা', 'কমিটি', 'আজ', 'রোববার', 'বিকেল', 'স্বাস্থ্য', 'দপ্তর',

'জিত', 'সংবাদ', 'ব্রিফিং', 'তথ্য', 'জানানো', 'আইইডিসিআর', 'পরিচালক', 'অধ্যাপক', 'মীরজাদী', 'সেব্রিনা']

DBSRA has performed better than the two above and give only one incorrect (underlined) word in the output.

The output of Bangla lemmatization framework is shown below:

['ডেঙ্গু', 'মৃত্যু', '১৬৯', 'ঘটনা', 'খবর', 'পাওয়া', 'স্বাস্থ্য', 'দপ্তর', '৮০', 'মৃত্যু', 'ঘটনা', 'পর্যালোচনা', 'করা', 'তত্ত্ব', 'রোগ', 'নিয়ন্ত্রণ', 'গবেষণা', 'প্রতিষ্ঠান', 'আইইডিসিআর', 'পর্যালোচনা', '৪৭', 'মৃত্যু', 'ডেঙ্গু', 'হয়', 'বলা', 'নিশ্চিত', 'করা', 'পর্যালোচনা', 'কমিটি', 'আজ', 'রোববার', 'বিকেল', 'স্বাস্থ্য', 'দপ্তর' আয়োজন', 'সংবাদ', 'ব্রিফিং', 'তথ্য', 'জানানো', 'আইইডিসিআর', 'পরিচালক', 'অধ্যাপক', 'মীরজাদী', 'সেব্রিনা']

But the proposed Bangla Lemmatization Framework gives no incorrect word in the output.

Table V shows the performances of the three algorithms and the framework based on accuracy for the lemmatization of Bangla language. This table confirms that the Bangla Lemmatization Framework gives the best accuracy. It is mentionable that the unknown words do not work properly using the four algorithms. If we can develop our corpus by adding more root words, then the accuracy can be increased.

TABLE V. ACCURACY PERFORMANCE OF THE LEMMATIZATION

Name	Levenshtein Distance	Trie	DBSRA	Bangla Lemmatization
Words	16132	16132	16132	16132
Correct	14459	14799	15022	15467
Incorrect	1673	1333	1110	665
Accuracy (%)	89.63	91.74	93.12	95.89

V. CONCLUSION

In this paper, we have presented three commonly used lemmatization algorithms and described their limitations of high response time in detail. Trie and Levenshtein Distance were not good for unknown words. DBSRA didn't work when lemma words were with other Bangla characters, which is not in inserted words. This is not a usual rule-based mechanism, but it is based on the mapping concept. We have fused all three algorithms to overcome those limitations by developing the Bangla Lemmatization framework. In this study, we found that this algorithm can't perform better with a smaller number of the root word. In the future, we will focus on detailed experimentations to increase accuracy. Instead of classical techniques, we will also concentrate on neural-based intelligent approaches.

REFERENCES

[1] Attia, M., Samih, Y., Shaalan, K., & Van Genabith, J. (2012, December). The floating Arabic dictionary: an automatic method for updating a lexical database through the detection and lemmatization of unknown words. In Proceedings of COLING 2012 (pp. 83-96).

[2] Mambrini, Francesco, and Marco Passarotti. "Harmonizing Different Lemmatization Strategies for Building a Knowledge Base of Linguistic Resources for Latin." Proceedings of the 13th Linguistic Annotation Workshop. 2019.

[3] Yang, W., Fang, Z., & Hui, L. (2016, July). Study of an improved text filter algorithm based on trie tree. In 2016 International Symposium on Computer, Consumer and Control (IS3C) (pp. 594-597). IEEE.

[4] Chrupała, G. (2014, June). Normalizing tweets with edit scripts and recurrent neural embeddings. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) (pp. 680-686).

[5] Schmitt, Marine, and Matthieu Constant. "Neural Lemmatization of Multiword Expressions." Proceedings of the Joint Workshop on Multiword Expressions and WordNet (MWE-WN 2019). 2019.

[6] Namly, Driss, et al. "Improving Arabic Lemmatization Through a Lemmas Database and a Machine-Learning Technique." Recent Advances in NLP: The Case of Arabic Language. Springer, Cham, 2020. 81-100.

[7] Stankovic, Ranka, et al. "Machine Learning and Deep Neural Network-Based Lemmatization and Morphosyntactic Tagging for Serbian." Proceedings of The 12th Language Resources and Evaluation Conference. 2020.

[8] Chakrabarty, Abhisek, Akshay Chaturvedi, and Utpal Garain. "CNN-based Context Sensitive Lemmatization." Proceedings of the ACM India Joint International Conference on Data Science and Management of Data. 2019.

[9] Boudchiche, Mohamed, and Azzeddine Mazroui. "A hybrid approach for Arabic lemmatization." International Journal of Speech Technology 22.3 (2019): 563-573.

[10] Boudlal, Abderrahim, et al. "A Markovian approach for arabic root extraction." Int. Arab J. Inf. Technol. 8.1 (2011): 91-98.

[11] Al-Shammari, Eiman, and Jessica Lin. "A novel Arabic lemmatization algorithm." Proceedings of the second workshop on Analytics for noisy unstructured text data. 2008.

[12] Schmitt, Marine, and Matthieu Constant. "Neural Lemmatization of Multiword Expressions." Proceedings of the Joint Workshop on Multiword Expressions and WordNet (MWE-WN 2019). 2019.

[13] El-Shishtawy, Tarek, and Fatma El-Ghannam. "An accurate arabic root-based lemmatizer for information retrieval purposes." arXiv preprint arXiv:1203.3584 (2012).

[14] Bergmanis, Toms, and Sharon Goldwater. "Context sensitive neural lemmatization with lematus." Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). 2018.

[15] Müller, Thomas, et al. "Joint lemmatization and morphological tagging with lemming." Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. 2015.

[16] Korenius, Tuomo, et al. "Stemming and lemmatization in the clustering of finnish text documents." Proceedings of the thirteenth ACM international conference on Information and knowledge management. 2004.

[17] Gesmundo, Andrea, and Tanja Samardžić. "Lemmatisation as a tagging task." Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2. Association for Computational Linguistics, 2012.