# An Approach to Sort Unicode based Bengali Text using Trie

Ranit Debnath Akash
Department Computer Science and Engineering, Shahjalal University of Science and Technology, Sylhet

U. Khyoi Nu
Department of Computer Science and Engineering, Shahjalal University of Science and Technology, Sylhet

Biswapriyo Chakrabarty
Department of Computer Science and Engineering, Shahjalal University of Science and Technology, Sylhet

## ABSTRACT

This paper proposes a sorting algorithm for Unicode based Bangla texts using Trie.. Bengali texts can not be sorted using the Unicode character scheme as Unicode character sequence is different from the Bangla Academy character sequence. Moreover, Bengali, an Indo-Aryan language spoken by approximately 200 million people has some distinct properties with its diacritic signs. In this paper, we have sorted Bangla texts based on the Bangla Academy character order using an efficient information retrieval data structure. Our proposed algorithm is more memory efficient and is applicable to any unicode based Bangla text.

## General Terms

Theoretical Informatics.

## Keywords

Sorting Algorithm, Unicode Bengali text, Trie

## 1. INTRODUCTION

Bengali is one of the most spoken languages in the world. [1][2][3] It is ranked second in the Indian subcontinent and seventh in the world according to the population speaking in Bengali which is about 210 million covering 3.05% of the world population. Hence, it has become a crying need to standardize Bengali language such as Bengali keyboard layout, Bengali character recognition etc. As a very basic need of this standardization, sorting of Bengali words has become one of the demanding issues now a days. Few works has already been done on this topic. This paper proposes a new approach to sort Unicode Bengali texts using Trie. The approach is simple and easy to implement in any code. [4] Since Bangla Academy is national language authority of Bangladesh, the texts were sorted according to its standard.

## 2. BENGALI LANGUAGE

Bengali is one of the most used language in the word and it has a very complicated structure .

### 2.1 Base Letters

There are about 11 vowels and 39 consonants in Bengali alphabet known together as Base Letter.

**The Bangla vowels are**
অ আ ই ঈ উ ঊ ঋ এ ঐ ও ঔ
**The Bangla consonants are**
ক খ গ ঘ ঙ চ ছ জ ঝ ঞ ট ঠ ড ঢ ণ র্ ত থ দ ধ ন প ফ ব ভ ম য র ল শ ষ স হ ড় ঢ় য় ৎ ◌ং ◌ঃ ◌ঁ

### 2.2 Modifiers

Bengali modifiers can be categorized to two groups. One of them is vowel modifier and another one is consonant modifier.

#### *2.2.1 Vowel Modifiers*

The vowel modifiers are generally known as –কার. Out of 11 vowels, 10 are considered as modifiers.

**Table 1. Vowel Modifiers Example**

| Vowel | Vowel Modifier | Example |
|---|---|---|
| আ | ◌া | মা |
| ই | ◌ি | মি |
| ঈ | ◌ী | মী |
| উ | ◌ু | মু |
| ঊ | ◌ূ | মূ |
| ঋ | ◌ৃ | মৃ |
| এ | ◌ে | মে |
| ঐ | ◌ৈ | মৈ |
| ও | ◌ো | মো |
| ঔ | ◌ৌ | মৌ |

#### *2.2.2 Consonant Modifiers*

There are six consonant modifiers in Bengali language. They are called –ফলা.

**Table 2. Consonant Modifiers Example**

| Consonant | Consonant Modifier | Example |
|---|---|---|
| য | য-ফলা | ম্য |
| র | র-ফলা | ম্র |
| ন | ন-ফলা | ম্ন |

| | | |
|---|---|---|
| ল | ল-ফলা | ল্ল |
| ম | ম-ফলা | ম্ম |
| ব | ব-ফলা | ব্ব |

## 2.3 Compound Characters

Compound characters are some different types of character which are formed by combining two or more Bengali alphabets and it will act like a single one though built with more than one. There are about 285 compound characters in Bangla language[5]. Some of the examples of compound characters are:

**Table 3. Compound Characters Example**

| Bangla Word | Compound Character | Decompressed Form |
|---|---|---|
| ব্রাহ্মণ | হ্ম | হ + ্ + ম |
| যুক্ত | ক্ত | ক + ্ + ত |
| কষ্ট | ষ্ট | ষ + ্ + ট |
| মুগ্ধ | গ্ধ | গ + ্ + ধ |
| উদ্ধার | দ্ধ | দ + ্ + ধ |

## 3. RELATED WORKS

Aamira Shabnam et al.[6] proposed an easily Comprehendible Unicode Based Sorting and in which they have some drawback because they haven’t added any null modifier. Aamira Shabnam et. al.[7] also proposed a Faster Approach of sorting but their mapping and sorting order is different from the Bangla Academy standard.

Partha Sarathi Kar et al.[8] proposed an improved Unicode Based Sorting in which they have tried mapping every characters and compound letters which increases the memory or storage complexity.

Md. Mahfuzur Rahaman et al. [9] used a revised Unicode based sorting method with maintaining the Bangla Academy order including taking account of the ZWJ (Zero-Width-Joiner) and ZWNJ (Zero-Width-Non-Joiner) at the time of mapping and decompressing a word which gave a more correct sorting order than others regarding many situations.

## 4. DATA SET

We have used the corpus of বাংলা ডাটাসেট (কর্পাস) [10] to test our sorting method and it has given an excellent result.

## 5. METHODOLOGY

## 5.1. Behavior and assumptions

We assumed the followings

- Mapping is a must as Unicode character sequence doesn’t match the Bangla Academy character sequence. So we need to sort them according to Bangla Academy character sequence.
- As Md. Mahfuzur Rahaman et. al.(2016)[9] told, we have considered the ZWJ (Zero-Width-Joiner) and ZWNJ (Zero-Width-Non-Joiner) while mapping and also while decompressing a word.

### *5.1.1. Mapping*

We have used mapping as Md. Mahfuzur Rahaman et al. [9] suggested which is the best one till now.

**Table 4. Mapping method used in this method [9]**

| Unicode Value | Character | Mapped Value |
|---|---|---|
| 200C | ZWNJ | 00 |
| 200D | ZWJ | 01 |
| 0985 | অ | 02 |
| 0986 | আ | 03 |
| 0987 | ই | 04 |
| 0988 | ঈ | 05 |
| 0989 | উ | 06 |
| 098A | ঊ | 07 |
| 098B | ঋ | 08 |
| 098F | এ | 09 |
| 0990 | ঐ | 10 |
| 0993 | ও | 11 |
| 0994 | ঔ | 12 |
| 0982 | ং | 13 |
| 0983 | ঃ | 14 |
| 0981 | ঁ | 15 |
| 0995 | ক | 16 |
| 0996 | খ | 17 |
| 0997 | গ | 18 |
| 0998 | ঘ | 19 |

| | | |
|---|---|---|
| 0999 | ঙ | 20 |
| 099A | চ | 21 |
| 099B | ছ | 22 |
| 099C | জ | 23 |
| 099D | ঝ | 24 |
| 099E | ঞ | 25 |
| 099F | ট | 26 |
| 09A0 | ঠ | 27 |
| 09A1 | ড | 28 |
| 09DC | ড় | 29 |
| 09A2 | ঢ | 30 |
| 09DD | ঢ় | 31 |
| 09A3 | ণ | 32 |
| 09A4 | ত | 33 |
| 09CE | ৎ | 34 |
| 09A5 | থ | 35 |
| 09A6 | দ | 36 |
| 09A7 | ধ | 37 |
| 09A8 | ন | 38 |
| 09AA | প | 39 |
| 09AB | ফ | 40 |
| 09AC | ব | 41 |
| 09AD | ভ | 42 |
| 09AE | ম | 43 |
| 09AF | য | 44 |
| 09DF | য় | 45 |
| 09B0 | র | 46 |
| 09B2 | ল | 47 |
| 09B6 | শ | 48 |
| 09B7 | ষ | 49 |
| 09B8 | স | 50 |
| 09B9 | হ | 51 |
| 09BE | া | 52 |
| 09BF | ি | 53 |
| 09C0 | ী | 54 |
| 09C1 | ু | 55 |
| 09C2 | ূ | 56 |
| 09C3 | ৃ | 57 |
| 09C7 | ে | 58 |
| 09C8 | ৈ | 59 |
| 09CB | ো | 60 |
| 09CC | ৌ | 61 |
| 09CD | ্ | 62 |

## 5.2. String store and sort:

In this paper a new methodology is proposed which is a little different from the others. Here a Trie data structure is built with the Bangla letters which are mapped at first to some numeric values. Now with that Trie structure we are storing every word in our corpus in the Trie tree. Basically Trie is a rooted tree where initially there is only one node which is root node from where we are starting the Trie tree. So it is initially empty and it has only root node. Now for this data structure every node has

- About 63 links to its children all of which point to different Bengali characters.
- A Boolean field which tells if this node is end point of any inserted word.

We insert words in Trie by searching it in the tire. For example we start from the then we search a link that correspond to the first character of the word. Here could be two cases:

1. A link may exist. Then we go to the next child level node with that link and the algorithm continue its' search for next character of word.
2. There is no link with the corresponding character. Then we create a new node and link it with it's parent's link for the corresponding position matching words' current character. Then we repeat this step till we get to the last character of the inserting word, when we have created a similar node or pointed it somehow for the last character we set the Boolean field of that node as true because it is end of the inserting word.

Now when we are inserting a word in Trie e.g. the Bangla word "কলম " we will search for an edge from the root named ″ক″ to another node where we can reach from the root node to that with the specific letter ″ক″ let's say it's a node n1 if it exists then good otherwise we create as described above. In the same way from n1 we will search and go to another node with edge labeled 'ল' to node n2, then to another node n3 with edge labeled 'ম' & after reaching n3 we will mark its' Boolean field as true as it is the end of the word. Now we store them in Bangla Trie normally letter by letter. But we are making the edge of letters from one to another starting from root in which order they are appearing in the word. We can see an example for the word "আম″. Here deeply colored node means a word ends in that node.

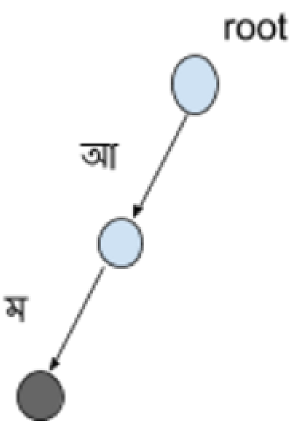


**Fig 1.1: Trie word store 1**

Now when we are storing the word ″আমি″ and "আমার" added in that structure it'll look something like below.

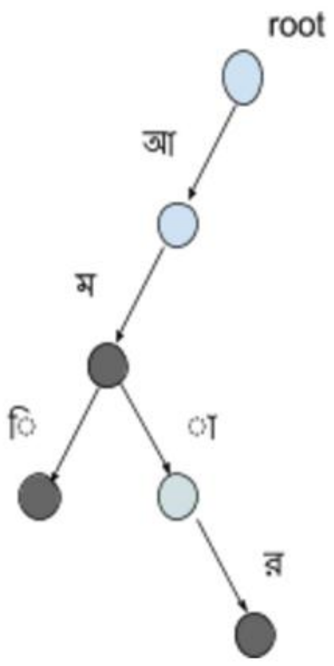


**Fig 1.2: Trie word store 2**

Now this is how Trie is storing words in the tree which is very memory efficient. For the same prefix it is not storing it twice. Now what we need to do after taking all the words we have to search it and print it in the sorted order. Now we will traverse the Trie starting from root node. We'll traverse it in the Bengali word letter order that means we'll first look for the letters in the order 'অ', "আ", ..., "ও", 'ঔ', 'ক', 'খ', ... ং, ঁ,া, ি, ী, ... , ো, ৌ etc in every node. That means after coming to a node it will search for if there is a edge from it labeled অ, then আ, then in the above order which is a defined order in Bengali letters. Then after finding a node it will do the same and will also check every node whether it ends with a word or not. If it's a word it will be printed and then it will look for another edge from that node. Now if we traverse the tree in the predefined order we will get a sorted list of Bengali words.

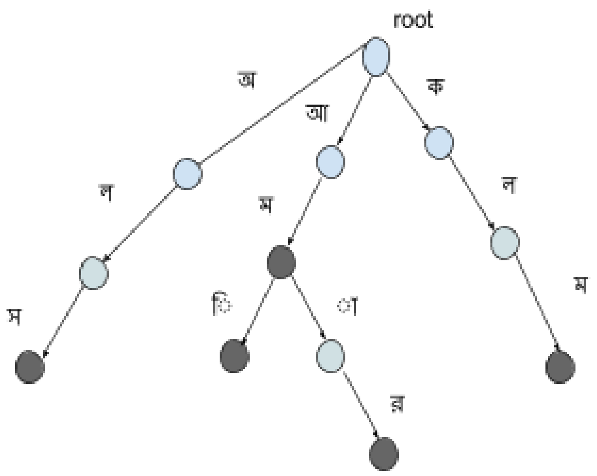


**Fig 1.3: Trie Data Structure**

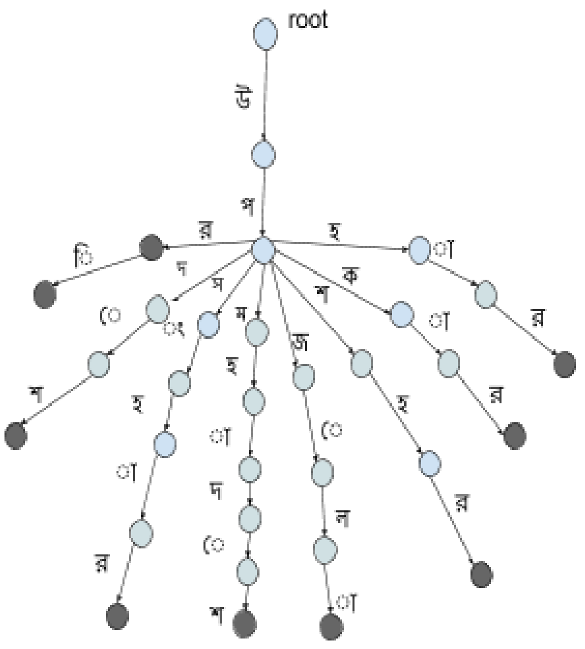
When it will traverse through Trie for the words in the above tree at first we will go for the 'অ' then from that we will again search and we will find a edge for 'ল' we will go to that then from there we will find the edge with 'স' then we will find an end mark in the that node so we will print the word 'অলস' then will traverse back and go to the root. Then from there we have traversed for 'অ' now from there it will traverse for 'আ' then from there it will go with the edge labeled 'ম' and it has an end mark so it will print the word 'আম' then from there it will go for ' া ' with then from there we will go to 'র' and it will have the end mark and print the 'আমার', then it will traverse back and now it go for the edge labeled 'ি' then it will go back to root then it will search and go for the edge labeled 'ক' then eventually it will discover the word 'কলম'. So in that way all the words will be printed in a sorted order.

## 6. RESULT AND ANALYSIS

Complexity of Inserting of a word in Trie: It has a time complexity of O(n), where m is the word length cause here In each iteration of the algorithm, we either search or create a node in the Trie till we reach the last of the word. This will take n effort. It also have a space complexity of O(n), because the worst case newly inserted key doesn't share a prefix with the the keys already inserted in the Trie. We have to add n new nodes, which takes us O(n) space.

Complexity of Traversing the Trie tree and printing it in sorted order: Here if we have total V vertices and total E edges in the Trie Tree then we have an complexity of O(V +E) to traverse the tree and print the words in the sorted order.

This method is memory efficient for the words with same prefix cause then we don't have to store all characters of the words, then it will share the same prefix characters then only add the extra characters that are not in its' shared prefix.



For example, when উপদেশ, উপকার, উপসংহার, উপমহাদেশ, উপরি, উপশহর, উপজেলা, উপরোক্ত, উপহার all share a common prefix উপ so we don't need to store the prefix more than once we will add the other characters of different words other than the common prefix after that.

Here our approach is obviously memory efficient than other methods and also time efficient. We hope that it can be taken as a new standard of sorting Bangla Unicode words.

## 7. CONCLUSION

The proposed algorithm sorts Unicode Bengali texts according to the character order of Bangla Academy successfully using Trie. This method is quite memory efficient and works pretty well for large dataset too. So this approach can be considered to be a standard to sort Unicode Bengali texts efficiently in accordance with Bangla Academy.

## 8. REFERENCES

[1] https://en.wikipedia.org/wiki/Bengali_language

[2] http://www.listsworld.com/top-10-languages-most-spoken-worldwide/

[3] http://timesofindia.indiatimes.com/india/Nearly-60-of-Indians-speak-a-language-other-than-Hindi/articleshow/36922157.cms

[4] https://en.wikipedia.org/wiki/Bangla_Academy

[5]http://forum.daffodilvarsity.edu.bd/index.php?topic=11714.0

[6] Aamira Shabnam, Debakar Shamanta Piklu, "An Easily Comprehendible Unicode Based Sorting Algorithm for Bangla Words"

[7] Aamira Shabnam, Tapashee Tabassum Urmi, Md. Saiful Islam, "A Faster Approach to Sort Unicode Represented Bengali Words"

[8] Partha Sarathi Kar, Shantanu Mandal, Labiba Jahan, "An Improved Unicode Based Sorting Algorithm for Bengali Words"

[9] Md. Mahfuzur Rahaman, "A Revised Unicode based Sorting Algorithm for Bengali Texts"

[10] বাংলা ডাটাসেট (কর্পাস) of http://scdnlab.com/corpus/