BANGLA SPELL CHECKER

AND

SUGGESTION GENERATOR

Sourav Saha
Student Id: 011133044
Faria Tabassum
Student Id: 011141114
Kowshik Saha
Student Id: 011142009
Marjana Akter
Student Id: 011142010

A thesis in the Department of Computer Science and Engineering presented
in partial fulfillment of the requirements for the Degree of
Bachelor of Science in Computer Science and Engineering

United International University
Dhaka, Bangladesh
December2018

Declaration

We, Kowshik Saha, Sourav Saha, Marjana Akter, Faria Tabassum declare that this thesis titled, Thesis Title and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a BSc degree at United International University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at United International University or any other institution, this has been clearly stated.
- Where we have consulted the published work of others, this is always clearly attributed.
- Where we have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely our own work.
- We have acknowledged all main sources of help.
- Where the thesis is based on work done by ourselves jointly with others, we have made clear exactly what was done by others and what we have contributed myself.

Kowshik Saha, 011142009, CSE

Sourav Saha, 011133044, CSE

Marjana Akter, 011142010, CSE

Faria Tabassum, 011141114, CSE

Certificate

I do hereby declare that the research works embodied in this thesis/project entitled "**Bangla Spell Checker And Suggestion Generator**" is the outcome of an original work carried out by Kowshik Saha, Sourav Saha, Marjana Akter, Faria Tabassum under my supervision.

I further certify that the dissertation meets the requirements and the standard for the degree of BSc in Computer Science and Engineering.

Dr. Mohammod Nurul Huda
Professor and Coordinator-MSCSE
United International University

Abstract

This thesis is mainly focused on finding out the misspelled words and providing the most optimized suggestion for Bengali words. The system that has been developed to make the path more ease of checking Bengali spelling can be used in any kind of system as an inherent integration which operates on Bengali language.

To produce an appropriate spell checker for Bengali language is always a great deal because of its complex nature and grammatical rules. It is common in Bengali language where many times the rule violates itself which is called exception.

Considering everything, we propose an algorithm which is a intelligent mixed up of some of the world famous algorithms like edit distance, Soundex, metafone along with string matching.

After implementing the already existed algorithms, blending it and using encoding to make Bangla acquainted to the algorithms, we will get the most probable suggestion for our misspelled Bengali words.

Acknowledgement

At the very outset, I would like to express my deep gratitude to Almighty who gave us enough knowledge and patience to complete the thesis within the stipulated time.

I would like to give special thanks my supervisor Prof. Dr. Mohammad Nurul Huda for his precious as well as constructive suggestions during the planning and development of this research work.

I would also like to extend my thanks to the concerned faculty member of United International University for their kind support throughout the research work.

Finally, I wish to thank my parents for their encouragement through my study.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Background

Bengali is one of the most widely spoken language as well as a common language in this Indian subcontinent. It is the need of time and situation to work with Bengali spelling and find out more optimized way so that the system can easily find out not only the misspelled Bengali words but also the most probable words which the user might have missed or wrote it wrongly.

The characteristics of Bengali is a little bit more complex which is an obstacle to rely on a single standalone solution. Besides the language has some orthographic rules which can create a large gap between the writing pattern of the word and the pronouncing pattern of the word.

Due to these reasons, there are some common type of errors such as non-word error, typing error and cognitive error. All these are result of phonetic similarity of Bengali characters, phonetic utterance difference.

To solve this issue we will describe in this thesis paper that how we can use different types of algorithm so that we can filter out most probable suggestion for the misspelled word as accurate as possible.

1.2 Previous Works

The field of working with Bengali spell checker started in 2002 when Hoque and Kaykobad keyboard was proposed which was basically a phonetic encoding for Bengali words. In 2004, Jaman and Khan proposed their new version of phonetic encoding. Both of the methods used Soundex mechanism but later one was an updated version. Some Bangla spell checking software has also been developed in small aspect. Mozilla Firefox browser has an add on which can check misspelled Bengali words which can work after installation.

In 2010 an add-on was developed for MS office called “Shuddhashabdo”. It is also a great job for Bengali spell checker.

In 2003 the most known Avro was developed by Omicron Lab which is basically a Bengali keyboard which also does the job of Bengali spell checking in a different angle.

1.3 Problems of previous works

The Haque and Kaykobad keyboard and the Jaman and Khan keyboard both have a similar lacking-both are based on basic Soundex algorithm.

Though it is true that the result rate derived by Soundex is very fast but the accuracy is very poor especially in complex case like name searching problem. Soundex solely depends on first letter, it also can’t capture phonetic similarity between words with silent consonants.

The Firefox add-on is only limited to use in Mozilla browser and also it has some version synchronization problem. The Shuddhashabdo spell checker can only be used in MS word. Besides it omits the most known spell checking mechanism like Metaphone, double Metaphone etc. The interface is also not user friendly as it is supposed to be.

1.4 Objective

In any kind of spell checker the main objective is supposed to recognize the word and provide automatic correction. These include word processors, machine translation, search engines and voice recognition.

Unlike other spoken languages, Bengali language still lacks an all in all spell checker which can do its job flawlessly. Hats why the aim of our project is to build an simple but efficient spell checker which will be robust, reliable and comprehensive.

The objectives of our Bengali spell checker is to detect misspelled words with indication, to find out the best probable correct words of the misspelled words, to replace the correct word with misspelled word without any hassle, to make the spell checker platform independent so that it can perform in any system as an built in process and to make the user interface absolutely simple.

Chapter 2

Algorithms for Spell Checker

2.1 String Matching

String matching is such a algorithm where one or several strings are tried to find out from the given text or database. It is a significant class of string algorithms that works to search a position where a single string or multiple strings can be found. Suppose, ***Q*** is a set of alphabet (Finite set). [2]

This ***Q*** symbol may either contain the capital alphabets {A,B,C,…..,Z} small letters as well or the binary character {0,1} or DNA alphabet ***Q={A,C,G,T}*** as example.

Classification of String Searching or Matching

The various Algorithms are defined according to the patterns of the strings. Suppose 'l' is the size of a pattern and '**L**' be the size of catchable string and ***A*=|*Q***| is the size of the letters.

NSSA=(Naïve String Search Algorithm)

RKA=(Rabin Karp Algorithm)

KMPA=(Knuth Morris Pratt Algorithm)

BMSSA=(Boyer Moore String Algorithm)

BA=(Bitap Algorithm)

TWSMA=(Two Way String Matching Algorithm)

BOM=(Backward Oracle Matching)

BNDM=(Backward Non deterministic Dawg Matching)

Name of Algorithm	**Pre-working Time**	**Matching time**	**Space**
NSSA	0	O(**L**l)	0
RKA	O(l)	avg O(**L** + l) worst O((**L**-l) l)	O(1)
KMPA	O(l)	O(**L**)	O(l)
BMSSA	O(l + **A**)	Best O(**L** / l) Worst O(**L**l)	O(**A**)
BA	O(l + **A**)	O(**L**l)	
TWSMA	O(l)	O(l + **A**)	O(1)
BOM	O(l)	O(**L**)	
BNDM	O(l)	O(**L**)	

Table 1: Time and space complexity of different String matching algorithms [2]

Among these algorithms BMSSA has been considered the standard algorithm for the sting matching.

Matching Strategy:

- Prefix matching(KMPA, BA/Shift-And)
- Suffix matching(BMSSA)
- Matching most efficient factor(BNDM,BOM)
- Others(NSSA, RKA)

2.2 Edit Distance

Edit distance is a category of such type algorithm which determine the dissimilarity between two or multiple strings or in other word it counts the minimum number of operations that are obvious to transform one word to another. Though there are number of methods to find out the edit distance between two strings, most of the time the Levenshtein distance is considered as edit distance algorithm. [1]

The known edit distance methods are:

- Levenshtein distance
- Damerau–Levenshtein distance
- Longest common subsequence (LCS)
- Hamming distance
- Jaro distance

Levenshtein distance

This distance measures how much minimum operations or single character edits are needed to change one string to another. Single character edits mean insertion, substitution or deletion. The Levenshtein distance between two words or strings ‘a’ and ‘b’ is given as a a mathematical formula. [4]

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

Here i is the length of string *a* and j the length of *b*.

- $d_{a,b}(i-1,j)+1$ corresponds to a deletion (from a to b).
- $d_{a,b}(i,j-1)+1$ corresponds to an insertion (from a to b).
- $d_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)}$ corresponds to a match or mismatch, depending on whether the respective symbols are the same.

- The Levenshtein distance has some upper and lower bounds. They are:
- It must be at least the two string's size difference.
- If the strings are equal, it is 0 and hamming distance plays as upper bound.
- It has the length of the string which has most length.
- It follows triangular inequality.

Damerau–Levenshtein distance:

By basic definition it is almost similar to Levenshtein distance but it includes transpositions in the operations that are possible. Besides it does the other edit operations-insertion, substitution or deletion. The Damerau–Levenshtein distance between two strings 'a' and 'b' is defined as a mathematical expression. [5]

$$d_{a,b}(i,j) = \min \begin{cases} 0 & \text{if } i=j=0 \\ d_{a,b}(i-1,j)+1 & \text{if } i>0 \\ d_{a,b}(i,j-1)+1 & \text{if } j>0 \\ d_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} & \text{if } i,j>0 \\ d_{a,b}(i-2,j-2)+1 & \text{if } i,j>1 \text{ and } a[i]=b[j-1] \text{ and } a[i-1]=b[j] \end{cases}$$

Here i is the length of string *a* and j the length of *b*.

- $d_{a,b}(i-1,j)+1$ corresponds to a deletion (from a to b).
- $d_{a,b}(i,j-1)+1$ corresponds to an insertion (from a to b).
- $d_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)}$ corresponds to a match or mismatch, depending on whether the respective symbols are the same.
- $d_{a,b}(i-2,j-2)+1$ corresponds to a transposition between two successive symbols.

Longest common subsequence:

The LCS finds out the common subsequences among the strings at the same which is also longest. By dynamic programming foundation, LCS can be found. [9]

If there are two sequences X=(x1,xa,x3,…..) and Y=(y1,y2,y3……) the longest common subsequence is found by following a formula:

$$LCS(X_i, Y_j) = \begin{cases} \emptyset & \text{if } i = 0 \text{ or } j = 0 \\ LCS(X_{i-1}, Y_{j-1}) \frown x_i & \text{if } x_i = y_j \\ \text{longest}\left(LCS(X_i, Y_{j-1}), LCS(X_{i-1}, Y_j)\right) & \text{if } x_i \neq y_j \end{cases}$$

Hamming distance:

Hamming distance finds out the position number in which the symbols are different of corresponding two strings of same length. [10]

It actually counts the required substitutions having minimum numbers. Various conditions such as non-negativity, indiscernible identity, triangle inequality are covered by hamming distance.

Jaro distance:

Jaro distance finds out the distance between two strings by using prefix. Two strings will be more similar if the Jaro distance is lower. [11]

The lower means the more similar. If the score is 1, it means that there is no similarity between two strings and if the score is 0, it means that the two strings are exactly same.

For two strings m1 and m2, Jaro similarity is:

$$sim_j = \begin{cases} 0 & \text{if } m = 0 \\ \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) & \text{otherwise} \end{cases}$$

Here s determines the string length . m determines the matching character numbers. t is number of half transpositions.

2.3 Soundex

Soundex is a phonetic algorithm which actually converts any kind of string into Soundex code. Robert C. Russell and Margaret King developed Soundex in order to encode the homophones into the same representation so that the nearest homophones can be matched when in spelling small difference happens. [3]

To do the job Soundex partitions all letters into several groups where each group is identified but an unique number. But the block which contains H, vowel-A,E,I,O,U and semivowels-W,Y is given the identification number 0 because the consonants actually matters when we actually pronounce any word. [6]

Considering this fact of phonology vowels are not considered at the time of creating Soundex code and also when the letters are categorized, the phonology or the way the letters are pronounced are only considered to categorize the letters and to create the block table

The standard Soundex algorithm is defined by the following table

Table 1 - Soundex Algorithm Groups	
Letter	**Code**
B, F, P, V	1
C, G, J, K, Q, S, X, Z	2
D, T	3
L	4
M, N	5
R	6
H, Y, W	(omitted)
A, E, I, O, U	(omitted)

Table 2: Standard Soundex table

There are also some rules beside this table which are must to obtain the Soundex code.

1. The Soundex code consists of some numbers which come from the table. Each letter represent a particular number but the first letter is not replaced with number. The first letter is retained.

2. All the occurrences of vowels and semivowels are dropped and replaced with 0. Later the final Soundex code is formed dropping all 0.

3. If two or more letters have same Soundex number and they sit side by side, only the first letter is retained and the others are dropped. Again if two or more same category constant is adjacent separated by any vowel or semi vowel, the vowel as well as later consonants are discarded. This rule is applicable for the first letter too.

4. If any word has so few letters that it is not possible to assign three numbers for the coding, we have to fill or append it with 0 until there are three numbers. Also if there are more than three number, we have to drop another numbers retaining the first three numbers because Soundex code should always have one beginning letter and three numbers-not more, not less.

2.4 Metaphone

Metaphone is a phonetic algorithm which is basically an improvement of Soundex algorithm based on English language. There are some constraints like variations and inconsistencies in English spelling. To solve the issue Metaphone is an improved one step for the production of more accurate encoding.

It is actually one category of fuzzy searching method where the Metaphone key is used to search each string in record. In 1990, Lawrence Philips published this phonetic algorithm and the main motive was to produce an approximate phonetic representation.

Metaphone reduces the alphabet into 16 constant letters 0BFHJKLMNPRSTWXY. [8]

Here the ‘0’ represents “th”, 'X' represents "sh" or "ch" and the others represent what they are supposed to represent.

These 16 letters are used for Metaphone encoding.

Unlike index, Metaphone follows some basic rules which define how Metaphone code will be created for every word. The below table defines the transformation, replacement, drop and all other necessary actions.

- Drop duplicate adjacent letters, except for C.
- If the word begins with 'KN', 'GN', 'PN', 'AE', 'WR', drop the first letter.
- Drop 'B' if after 'M' at the end of the word.
- 'C' transforms to 'X' if followed by 'IA' or 'H' (unless in latter case, it is part of '-SCH-', in which case it transforms to 'K'). 'C' transforms to 'S' if followed by 'I', 'E', or 'Y'. Otherwise, 'C' transforms to 'K'.
- 'D' transforms to 'J' if followed by 'GE', 'GY', or 'GI'. Otherwise, 'D' transforms to 'T'.
- Drop 'G' if followed by 'H' and 'H' is not at the end or before a vowel. Drop 'G' if followed by 'N' or 'NED' and is at the end.
- 'G' transforms to 'J' if before 'I', 'E', or 'Y', and it is not in 'GG'. Otherwise, 'G' transforms to 'K'.
- Drop 'H' if after vowel and not before a vowel.
- 'CK' transforms to 'K'.
- 'PH' transforms to 'F'.
- 'Q' transforms to 'K'.
- 'S' transforms to 'X' if followed by 'H', 'IO', or 'IA'.
- 'T' transforms to 'X' if followed by 'IA' or 'IO'. 'TH' transforms to '0'. Drop 'T' if followed by 'CH'.
- 'V' transforms to 'F'.
- 'WH' transforms to 'W' if at the beginning. Drop 'W' if not followed by a vowel.
- 'X' transforms to 'S' if at the beginning. Otherwise, 'X' transforms to 'KS'.
- Drop 'Y' if not followed by a vowel.
- 'Z' transforms to 'S'.
- Drop all vowels unless it is the beginning.

Chapter 3

Proposed Approach

3.1 Introduction

While proposing the encoding we have to keep in mind few things. We are working on Bengali language, a language for which no algorithm has been developed.

The complex approach of Bengali words like (YA fala, RA fala, MA fala, RA fala), inner grammatical rules like sound theory, vowel consonant coalition are great obstacles to find an standalone solution for Bengali suggestion generator. [7]

That's why we make an assimilation of some great phonetic algorithms Soundex and Metaphone where algorithms are based on phonology along with Fuzzy matching approach-Levenshtein Distance and basic String matching algorithm.

Phonetic encoding uses phonology as its algorithm is built up to handle the complexities and difficulties which might arrive for any kind of spell checking and also for suggestion giving. Besides phonetic algorithms Soundex and Metaphone, we have also used Edit Distance algortihm-Levenshtein distance because the basic phonetic algorithms were designed to work on English language.

Though the phonetic algorithms work on Bengali Word pretty well but by giving another layer filtering of edit distance algorithm will increase the accuracy as it produces values depending on their similarity and closeness.

Again, for errors like repeated letters, transposed letters or hitting the wrong key-edit distance algorithm becomes a great choice.

At the beginning of process of whether the word is misspelled or not-the answer is given by String matching algorithm. It just cross checks and gives direction to what the rest algorithm will do.

Though we have used String Matching algorithm to see the input word is available is database or not, by the increase of database the check process will occur more time complexity.

As a solution, we have to use Morphological parser which will help us to reduce the database and not to increase time complexity. Morphological parser is a process where root word is determined. It is actually a method of natural language processing

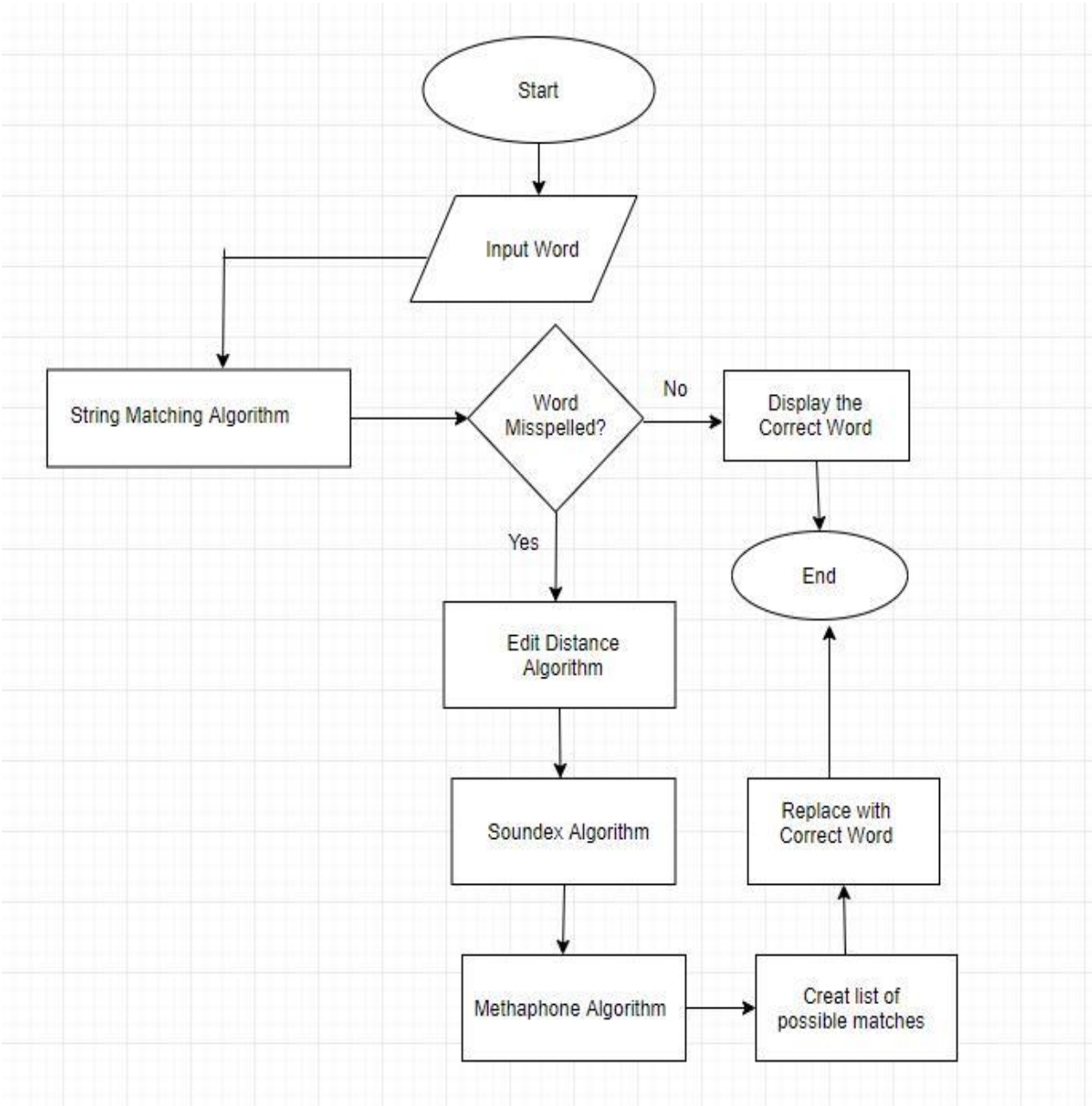
As stated earlier, the basic algorithms are English language based. So we presented every Bengali word as English, did the process job and then again converted the English words in Bengali. This conversion is not translation.

Rather, here Bengali words are written using English symbols so that the phonetic algorithm considers the Bengali words as English. To do that, we have converted 50 Bengali letters with 26 English letters. The list is given below-

অ=Oo	আ=A	ঈ=II	ই=I	
ঊ=UU	উ=U	ঋ=RRI		
ঐ=OI	এ=E	ঔ=OU	ও=O	
া=a	ি=i	ী=ii	ু=u	ূ=uu
ৃ=rri	ে=e	ৈ=oi	োা=o	ৌা=ou

খ=kh	ক=K	ঘ=gh	গ=G	ঙ=ng
ছ=ch	চ=C	ঝ=jh	জ=J	ঞ=NG
ঠ=Th	ট=T	ঢ=Dh	ড= D	ণ=n
থ=th	ত=t	ধ=dh	দ=d	ন=N
ফ=ph	প=P	ভ=bh	ব=B	ম=m
য=Z	র=r	ল=L		
শ=sh	ষ=S	স=s	হ=h	
ঢ়=Rh	ড়=R	য়=Y	ং =Ng	

3.2 Flowchart



.

3.3 Algorithm

1. Start
2. Declare variables and convert Bengali letters into English letters
3. Use String matching algorithm function to check the availability input words in database
4. If

   Input word available in database

   Show the word as it is
5. Else

   Use multiple functions to convert Bengali word into English phonetic word. Go through every word of database and use edit distance algorithm to compare with input word pick top 30 least words having least edit distance with input word run Soundex algorithm with the selected words and find shortlisted words
6. If

   No word available, show the words found by edit distance algorithm
7. Else

   Run Metaphone algorithm with the selected words and find shortlisted words
8. If

   No word available, show the words found by Soundex algorithm
9. Else

   Show words found by Metaphone algorithm
10. Stop

Chapter 4

Experiment

4.1 Corpus

We have designed a database of Bengali words which are the most common misspelled words.

Our proposed system runs on the base of this database. So we have filled our database with more than 2500 Bengali words in sql file as we have done our project in php.

4.2 Experimental Setup

To do the experiment of finding out misspelled words in any kind of Bengali writing and give proper suggestions for the misspelled words, we have designed a new method using some existing algorithms.

To set up the environment of the algorithms, we have used php language along with HTML-CSS for the visual purpose and JavaScript for interactive response. We have used three types of algorithms.

They are String matching algorithm, Edit distance algorithm and Phonetic algorithm.

For string matching, we have used NSSA or naive string matching algorithm.

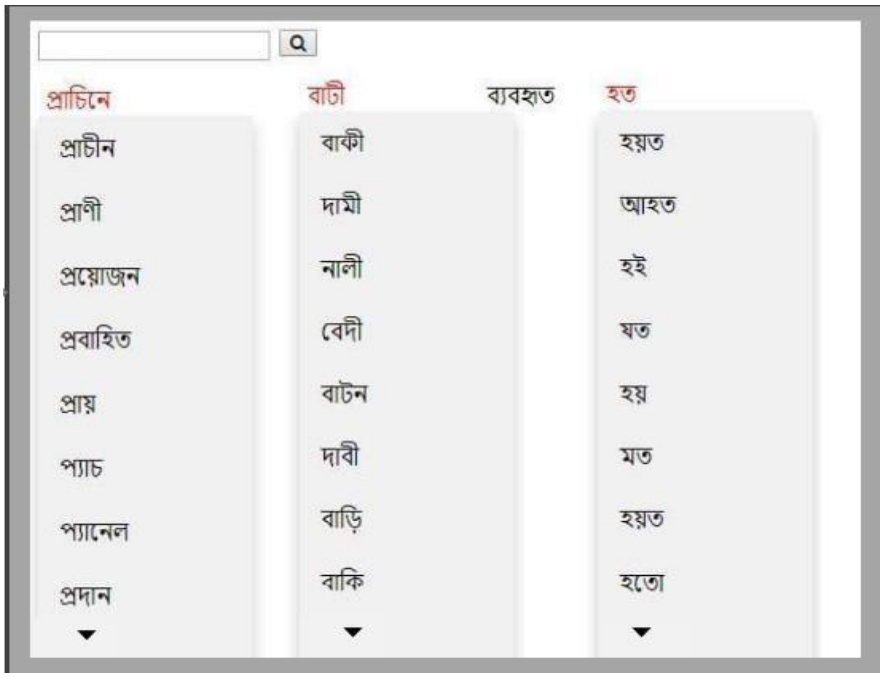
For edit distance we have used Levenshtein edit distance algorithm.

For phonetic we have used Soundex and Metaphone phonetic algorithm.

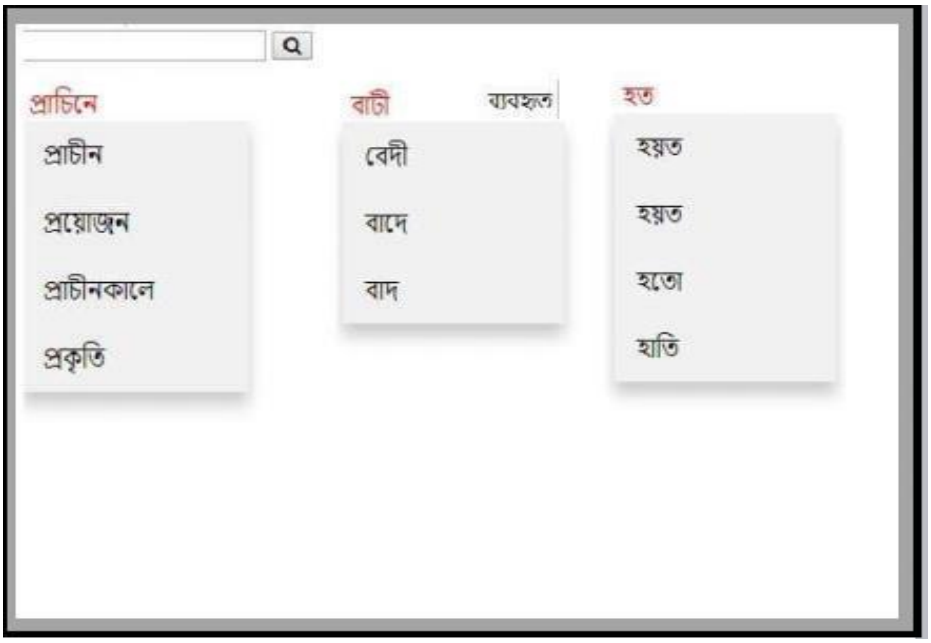
4.3 Experimental Result

In case of string matching we got almost 99% accuracy of finding out the misspelled words and identify them correctly.

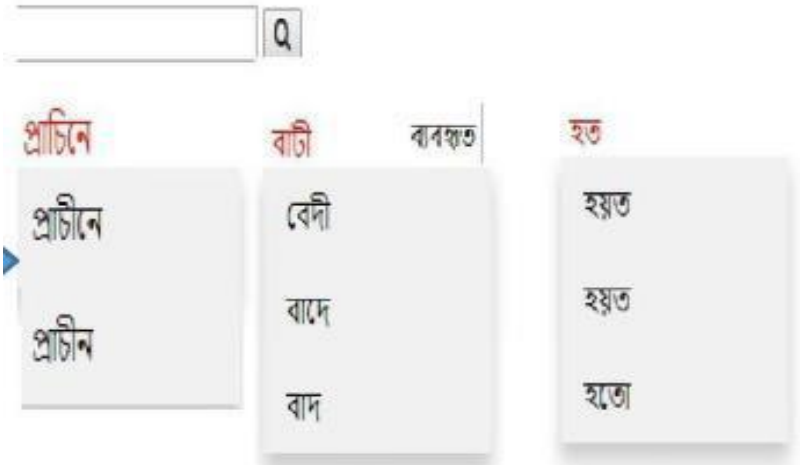
In terms of edit distance after using string matching, we got about 50% accuracy. It finds the appropriate misspelled words along with some words which are not relevant at all.



After using Soundex along with String matching and edit distance, the accurate boosts up interestingly. By using different types of examples of misspelled Bengali words, we got about 80% accuracy.



Finally after using Metaphone on previous result what we found using String matching, Edit distance and Soundex algorithm, we also get more accurate suggestion. Though for many examples of ligature, it doesn't give accurate suggestions. That's why finally we get about 85% accuracy overall.



Chapter 5

Conclusion and Future Work

5.1 Conclusion

The improvement of Bengali spell checker and suggestion generator is suggested by using a noncomplex method without proposing any brand new algorithm but by using the existing algorithms in an effective way.

So that we can extract the best feature of every algorithm- string matching, edit distance, Soundex and Metaphone and fill-up the lacking by using other algorithms.

We discussed the basic introduction of these algorithms and showed the method of using in a systematic way so that the accuracy of suggestion generation improves step by step.

The summary regarding the features of our proposed system is-

1. Crosschecking each and every word and identify Misspelled words.
2. Top accurate suggestions filtered by 3 algorithms and replacement with one of the mini listed words.

It can be used as a plugin in applications like WPS office or Microsoft office as well ws can be integrated in an system that supports Bengali word to help in name searching activities and to find out errors.

It is also feasible in the field where multiple suggestions are needed in ranking.

5.2 Future Work

We will try to upgrade our system which will be able to do the following works:

1. Integrate the spelling checker into Office packages like Open Office and MS Office.
2. Adding more words in database and enrich it with the latest Bangla words.
3. Develop the system by integrating more advanced phonetic algorithms like double Metaphone and Metaphone 3.

References

[1] N.UzZaman, “Phonetic Encoding for Bangla and its Application to Spelling checker, Name searching, Transliteration and Cross language information retrieval”, Undergraduate thesis (Computer Science), BRAC University, May 2005.

[2] String searching algorithm available online at
https://en.wikipedia.org/wiki/String-searching_algorithm

[3] Naushad UzZaman and Mumit Khan “A Bangla Phonetic Encoding for Better Spelling Suggestions”, Proc. 7th International Conference on Computer and Information Technology, Dhaka, Bangladesh, December, 2004.

[4] Levenshtein edit distance algorithm, available online at
https://en.wikipedia.org/wiki/Levenshtein_distance

[5] Damreu-Levenshtein distance available online at
https://en.wikipedia.org/wiki/Damerau%E2%80%93Levenshtein_distance

[6]The Soundex Algorithm, available online at
https://en.wikipedia.org/wiki/Soundex

[7] N. UzZaman and M. Khan, “A Comprehensive Bangla Spelling Checker”,
Center for Research on Bangla Language Processing, BRAC University, Bangladesh.
Supported in part by the PAN Localization Project (www.panl10n.net), grant from the International Development Research Center, Ottawa, Canada.

[8]The Metaphone Algorithm, available online at
https://en.wikipedia.org/wiki/Metaphone

[9] Longest common subsequence available online at
https://en.wikipedia.org/wiki/Longest_common_subsequence_problem

[10] Hamming distance available online at
https://sciencing.com/how-to-calculate-hamming-distance-12751770.html

[11] Jaro distance available online at
https://en.wikipedia.org/wiki/Jaro%E2%80%93Winkler_distance

Appendix A

Code

```
<?php
include 'dbh.php';
$bengal=array('অ','আ','ঈ','ই','ঊ','উ','ঋ','ঐ','এ','ঔ','ও','া','া','া','া','া','া','া','া','া','
া',
        'খ','ক','ঘ','গ','ঙ','ছ','চ','ঝ','জ','ঞ','ঠ','ট','ঢ','ড','ণ','থ','ত','ধ','দ','ন',
        'ফ','প','ভ','ব','ম','য','র','ল','শ','ষ','স','হ','ঢ়','ড়','য়','া');
$english=array('Oo','A','II','I','UU','U','rri','OI','E','OU','O','a','ii','i','uu','u','rri','e','oi', 'ou','o',
        'kh','K','gh','G','ng','ch','C','jh','J','NG','Th','T','Dh','D','N','th','t','dh','D','N',
         'ph','P','bh','B','m','Z','r','L','sh','S','s','h','Rh','R','Y','ng');
if(isset($_POST['bubu']))
{
$str2=$_POST['search'];
$word2=explode(" ",$str2);
$str=str_replace($bengal, $english, $str2);
$word=explode(" ",$str);

 foreach ($word as $value)
  {
     $flag=0;
     $sql="SELECT * FROM id";
     $result = $conn->query($sql);
     while ($row = $result->fetch_assoc(){
    if(strcmp($value,str_replace($bengal, $english,$row['text']))==0)
       {

        $flag =1;
       }
     }
     if ($flag==1)
```

```
    {
      echo str_replace($english, $bengal,$value); echo "
      ";
    }
    else
    {
        $abs=array();
        $abs2=array();
        echo "<font class='a'>";
        //echo $value;
?>
        <div class="dropdown">
  <button class="dropbtn"><?php echo str_replace($english, $bengal,$value); ?></button>
  <div class="dropdown-content">

<?php
        echo " ";
        echo "</font>";
    }
    if($flag==0)

   {
     $sql1="SELECT * FROM id";
     $result1 = $conn->query($sql1);
```

```
 while($row1=$result1->fetch_assoc())
 {
 array_push($abs,bangla($value,str_replace($bengal, $english,$row1['text'])));
  array_push($abs2,str_replace($bengal, $english,$row1['text']));
 }
asort($abs);
$f=array_keys($abs);
$Soundex=array();
for($i=0;$i<30;$i++)
array_push($Soundex,$abs2[$f[$i]]);

$metafn=array();
foreach ($Soundex as $boo)
if(strcmp(Soundex($boo),Soundex($value))==0)
array_push($metafn, $boo);

 $ans=array();
 foreach($metafn as $boo)
 array_push($ans,str_replace($english, $bengal,$boo));

foreach($ans as $boo)
{
?>
<a onclick="correctWord(this)" name="<?php echo $boo ?>" > <?php echo $boo ?> </a>

  <?php
 }
 ?>
```

```
  </div>
        <script>

           function correctWord(val){

         var a = val.parentNode.parentNode;

             a.    getElementsByTagName("button")[0].innerHTML=val.name;

              a.getElementsByTagName("button")[0].style.color="black";
            }

        </script>
</div>
<?php
function minx($a, $b, $c)
{
  if ($a <= $b && $a <= $c) {
    return $a;
  }
 else if ($b <= $a && $b <= $c) {
    return $b;
  }
else
{

  return $c;
}
function bangla($a, $b)
```

```
{
  $aLength = strlen($a);
  $bLength = strlen($b);
  //$dp[$aLength + 1][$bLength + 1];

  for ($i = 0; $i <= $aLength; ++$i) {
    $dp[$i][0] = $i;
  }
  for ($j = 0; $j <= $bLength; ++$j) {
    $dp[0][$j] = $j;
  }

  for ($i = 1; $i <= $aLength; ++$i) {
    for ($j = 1; $j <= $bLength; ++$j) {
      if ($a[$i - 1] == $b[$j - 1]) {
        $dp[$i][$j] = $dp[$i - 1][$j - 1];
      }
      else {
        $dp[$i][$j] = 1 + minx($dp[$i][$j - 1],$dp[$i - 1][$j],$dp[$i - 1][$j - 1]);
      }
    }
  }
  return $dp[$aLength][$bLength];
```