

Towards Comprehensive Bangla Computing: Corpus and Lexicon with Spell and Grammar Checker

Nahid Hossain
Student Id: 012163016

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



United International University

Dhaka, Bangladesh

February 2019

©Nahid Hossain, 2019

Certificate

This thesis titled “*Towards Comprehensive Bangla Computing: Corpus and Lexicon with Spell and Grammar Checker*” submitted by Nahid Hossain, Student ID: 012 163 016, has been accepted as Satisfactory in fulfillment of the requirement for the degree of Master of Science in Computer Science and Engineering on 2nd February 2019.

Board of Examiners

1. _____
Dr. Mohammad Nurul Huda
Professor and MSCSE Program Director
Department of Computer Science and Engineering (CSE)
United International University (UIU)
Dhaka-1212, Bangladesh
Supervisor
2. _____
Dr. Md. Abul Kashem Mia
Dean and Professor
School of Science and Engineering
United International University (UIU)
Dhaka-1212, Bangladesh
Head Examiner
3. _____
Dr. Swakkar Shatabda
Associate Professor and BSCSE Program Coordinator
Department of Computer Science and Engineering (CSE)
United International University (UIU)
Dhaka-1212, Bangladesh
Examiner-I
4. _____
Rubaiya Rahtin Khan
Assitant Professor
Department of Computer Science and Engineering (CSE)
United International University (UIU)
Dhaka-1212, Bangladesh
Examiner-II
5. _____
Dr. Salekul Islam
Professor and Head of the Department
Department of Computer Science and Engineering (CSE)
United International University (UIU)
Dhaka-1212, Bangladesh
Ex-Officio

Declarations

I, Nahid Hossain, declare that this thesis titled, “*Towards Comprehensive Bangla Computing: Corpus and Lexicon with Spell and Grammar*” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for an MSc 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 I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Nahid Hossain

ID: 012163016

MSCSE Student, Department of Computer Science and Engineering

United International University

Dhaka, Bangladesh

I do hereby declare that the research works embodied in this thesis entitled “*Towards Comprehensive Bangla Computing: Corpus and Lexicon with Spell and Grammar*” is the outcome of an original work carried out by Nahid Hossain under my supervision.

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

Dr. Mohammad Nurul Huda

Professor and Director of MSCSE Program

Department of Computer Science and Engineering

United International University

Dhaka, Bangladesh

Abstract

In this thesis, we have presented a comprehensive Bangla spell and grammar checker and its building techniques. To make the grammar checker highly accurate and robust, we have built the largest Bangla monolingual corpus comprising over 100 million words. Moreover, we have built the largest Bangla lexicon with over 1 million unique words extracted from the monolingual corpus to enrich the spell checker. Since we have embedded immense data, to increase efficiency and to reduce processing time we have used hashing technique, pre-defined double metaphone and pre-defined counts for language model probability. In addition, our spell and grammar checker improves itself day by day by keeping an individual local log of an user's previous suggestions and choices for future suggestions which gives customized user experiences.

Bangla is a language spoken of over 300 million total speakers around the world. To write a fruitful Bangla article for diverse publications, it is highly required to have a robust spell and grammar checker. However, few pieces of research have been done on this side of language processing and therefore, in Bangla, there is no such spell and grammar checker which can provide highly appreciable output. Some studies have been done individually on the spell checker or grammar checker. However, checking spell and checking grammar at the same time is very essential for a novel article. Moreover, no researches have been done with such an immense amount of data like we did. In addition, all of these studies have been done only for research purposes without discerning practical applications. That is why all of these studies show several imperfections in performing real-life text processing. In this thesis, we have demonstrated the technique to build our corpus, lexicon, spell and grammar checker with describing the limitations of other studies and solution of those limitations.

Acknowledgement

I would like to start by expressing my deepest gratitude to the Almighty Allah for giving me the ability and the strength to finish the research successfully within the scheduled time.

I would also like to thank my honorable supervisor Dr. Mohammad Nurul Huda, Professor and MSCSE Program Director of United International University for his generous guidance during the whole research.

I am very much fortunate that I have received sincere guidance from Dr. Abul Kashem Mia, Dean, School of Science and Engineering, United International University, Dr. Swakkhar Shatabda, Associate Professor & Undergraduate Program Coordinator, United International University and Mrs. Rubaiya Rahtin Khan, Assistant Professor, United International University. I would like to express my gratitude to Dr. Salekul Islam, Professor and Head, Computer Science and Engineering Department, United International University for his encouragement during the whole work.

Finally, I would like to thank my parents for all the love and guidance.

Table of Contents

LIST OF TABLES	viii
LIST OF FIGURES.....	ix
1. Overview.....	1
1.1 Introduction.....	1
1.2 Previous Works	1
1.3 Problem Statements and Motivation	2
1.4 Objective and Applications.....	2
1.5 Organization.....	3
2. Corpus and Lexicon Creation	4
2.1 Introduction.....	4
2.2 Limitation of Previous Approaches.....	4
2.3 Creating Web Crawler and Scraper.....	5
2.4 Corpus Creation.....	8
2.5 Lexicon Creation	10
2.6 Distributing Lexicon with Pre-defined Double Metaphone Values.....	12
2.7 Sampling, Biasness and Strength	13
3. Spell Checker	14
3.1 Introduction.....	14
3.1.1 Soundex.....	14
3.1.2 Metaphone.....	15
3.1.3 Double Metaphone.....	15
3.1.4 Metaphone 3	15

3.1.5 Edit Distance	16
3.2 Types of Spelling Errors	17
3.2.1 Typographic Error	17
3.2.2 Cognitive Error.....	17
3.2.3 Visual Error	18
3.2.4 Space Related Word.....	19
3.3 Limitation of Previous Approaches and Solution	19
3.3.1 Trivial Lexicon size	19
3.3.2 Primitive Approaches	20
3.3.3 No Hashing.....	20
3.3.4. Not Suitable For Big Data Processing	21
3.3.5 Handling Foreign Words and Numbers	21
3.3.6 No Association of Spell Checker with Grammar Checker	22
3.3.7 No Customized User Experience.....	22
3.4 Our Double Metaphone Encoding.....	23
3.5 Algorithm and System Diagram of Spell Checker Section	25
4. Grammar Checker	28
4.1 Introduction.....	28
4.2 Limitation of Previous Approaches.....	28
4.2.1 Minor Corpus	28
4.2.2 Low Accuracy	29
4.2.3 Finding an error in complex, compound sentence structure	29
4.2.4 POS Based n-gram model	29
4.2.5 No Suggestion for Grammatical Error.....	30
4.3 Algorithm and System Diagram of Grammar Checker	31

5. Experimental Results and Analysis.....	35
5.1 Spell Checker System.....	35
5.2 Grammar Checker System.....	39
5.3 Performance Calculation.....	42
6. Conclusion and Future Work.....	44
7. References.....	45

LIST OF TABLES

Table 1 : Abstract configuration of the device used in scraping	8
Table 2 : Comparison all Bangla Monolingual text corpus.....	10
Table 3 : Sample result of N. UzZaman and Mumit's Bangla soundex table	15
Table 4 : Levenshtein's distance between two words.....	16
Table 5 : Levenshtein's distance between to double metaphone encoded words.....	16
Table 6 : Typographic error example in Bangla.....	17
Table 7 : Cognitive error example in Bangla	18
Table 8 : Almost similar Bangla alphabets.....	18
Table 9 : Visual error example in Bangla with International Phonetic Alphabet(IPA)	18
Table 10 : Space related errors example in Bangla.....	19
Table 11 : Metaphone values of 'ঋ' (VOCALIC R)	23
Table 12 : Example comparison of 'ঋ' between previous and our approach	23
Table 13 : Metaphone values of 'ঊ' (Vowel Sign Vocalic R)	24
Table 14 : Comparison of coding 'ঊ' between previous and our approach	24
Table 15 : Unicode and name correction for letter 'ঋ'	25
Table 16 : Table of terminology (spell checker).....	25
Table 17 : Comparison of POS based language model probability	30
Table 18 : Table of terminology (grammar checker section)	31
Table 19 : Test dataset size	42
Table 20 : Performance summary of our system	43

LIST OF FIGURES

Diagram 1 : System diagram of spell checker section	26
Diagram 2 : System diagram of grammar checker section	32
Figure 1 : Our spider crawling and scraping on prothomalo.com	9
Figure 2 : Input has no spell error	35
Figure 3 : Sentence with two spelling errors	35
Figure 4 : Sentence with two spelling errors and suggestion	36
Figure 5 : List of suggestions where tentative correct word not at the top	36
Figure 6 : Tentative correct word at the top based on previous log	37
Figure 7 : Foreign words handling	38
Figure 8 : Example of word ‘১০০০’ which is purely numerical	38
Figure 9 : Example of word ‘১০টি’ which is alphanumerical and incorrect	38
Figure 10 : Suggestion for the incorrect alphanumerical number.	39
Figure 11 : Detecting consecutive redundant words	39
Figure 12 : Suggestion for the consecutive redundant words error	40
Figure 13 : Detecting additional whitespaces and Suggestion to remove	40
Figure 14 : Identifying punctuation error(extra)	41
Figure 15 : Identifying punctuation error(missing)	41
Figure 16 : Identifying and suggesting a grammatically incorrect sentence(example-1) .	41
Figure 17 : Identifying and suggesting a grammatically incorrect sentence(example-2) .	42
Figure 18 : Accuracy and Error rate presentation	43

Chapter 1

Overview

1.1 Introduction

Bangla is a language vocalized by over 300 million native speakers around the world [1]. Every day thousands of books, magazines and newspapers are publishing in Bangla. In order to inscribe a high-quality article, an article must be free of spelling and grammatical inaccuracies. Consequently, spell and grammar checker is very essential for writing a fruitful article. Although there are numerous kinds of research have been done on spell and grammar checkers in English, a very few researches have been done in Bangla. We have spent ample time on studying the available Bangla spell and grammar checker and have figured out several boundaries in the current spell and grammar checker approaches. In this thesis, we have demonstrated all limitations and have claimed to solve all these confines.

1.2 Previous Works

In 2001, B.B. Chaudhuri proposed a reverse order dictionary based spell checker [2]. He mentioned phonetic problems in Bangla and tried to solve it by placing the same code to phonetically similar alphabets. M.T. Haque and M. Kaykobad proposed Bangla phonetic encoding for spell checker based on soundex algorithm in 2002 [3, 19]. They numbered each group of phonetically similar consonant characters with 1 to 9 and 0 for vowels. However, this Bangla soundex table shows several imperfections. Later in 2005, N. UzZaman and Mumit Khan projected a phonetic encoding based on soundex algorithm which abrogated several limitations of Haque and Kaykobad's encoding [4]. N. UzZaman and Mumit Khan proposed another spell checker based on double metaphone algorithm in 2005 [5]. In that paper, the authors proposed a double metaphone encoding technique which improves spell checker significantly. Later in 2006, the authors proposed a spell checker based on their previous approach, claiming a better suggestion of misspelled words [14]. In 2016, Sajid Ahmed et al. have used Edit Distance and n-gram

models to post-process their Optical Character Recognition (OCR) output by omitting misspelt words [22].

On the other hand, a very few researches have done on Bangla grammar checker. The only notable research on Bangla grammar checker was done by Md. Jahangir Alam et al. in 2006 [17]. In that paper, the authors presented an N-gram based statistical grammar checker for both Bangla and English. The system assigns tags to each word of a sentence using a POS tagger. Using gram analysis, the system determines a probability of tag sequence. However, the system has several limitations (described in Chapter 4) and the accuracy is as low as 53.7%. In 2011, K. M. Azharul Hasan et al. have projected a Context Free Grammar (CFG) for Bangla language and hence based on that grammar, they developed a parse tree [18]. In 2018, Sadidul Islam et al. constructed a standard benchmark dataset to build a Bangla grammar checking technique with the accuracy of 79% based on sequence-to-sequence recurrent neural network [21]. According to the authors, this parser can be applied for Bangla grammar checker to parse different Bangla sentences.

1.3 Problem Statements and Motivation

In order to write an impressive Bangla article, an article must be free from any spelling and grammatical blunders. However, there is no such efficient spell and grammar checker available in Bangla. Most of these approaches are primitive and based on very trivial datasets which ultimately led the systems into lower accuracy. Moreover, these approaches are not suitable for big data processing due to its primitive techniques of data analysis. Most importantly, all these approaches are either on spell checker or in grammar checker. No combination of both spell and grammar checker exists. However, we know the combination of both spell and grammar checker is highly important for a robust system. After studying on several aspects and the confines of Bangla spell and grammar checker and keen necessity of a highly efficient Bangla spell and grammar checker motivated us to design and develop a comprehensive Bangla spell and grammar checker.

1.4 Objective and Applications

The objectives of our research are the following

- 1) To develop a comprehensive and highly accurate Bangla spell and grammar checker.
- 2) To develop the most well-organized and the largest Bangla monolingual corpus and then to develop a Bangla monolingual lexicon by extracting the corpus.
- 3) To provide a customized user experience by keeping a local log of particular user's activity and a user-friendly interface.
- 4) To develop the system considering all real-life applications and situations which was not considered in most of the previous approaches resulting lower efficiency.
- 5) To boost up the processing speed for large input data by using the latest phonetic encoding algorithms and fastest data processing techniques such as using hashing and pre-processed data.

As we have mentioned earlier, to write a fruitful Bangla article for diverse publications it is essential to have a highly efficient Bangla spell and grammar checker. There are thousands of Bangla articles are publishing every single day in online and printed newspapers. In addition, thousands of books have been published every year by hundreds of publications. In social media such as Facebook and Twitter, millions of users are posting their status, news and announcements in Bangla every day. Moreover, there are many government and non-government places where documents are written in Bangla. In all the aforementioned places our proposed system would improve the article quality significantly.

1.5 Organization

Chapter 2 demonstrates every aspect of our monolingual corpus development. In this chapter, we have also explained how we have extracted and developed our monolingual lexicon. In Chapter 3, we have explained the spell checker section of our proposed system with limitations of previous approaches, system diagram and algorithm. On the other hand, the grammar checker section has been demonstrated in Chapter 4 with all the necessary details. We kept the details of the spell checker and grammar checker system in different chapters for simplicity and make the understanding of both systems easier. In Chapter 5, we have demonstrated the experimental results and analysis of the results. Finally, in Chapter 6 we have concluded our proposed system and explained our future work.

Chapter 2

Corpus and Lexicon Creation

2.1 Introduction

A text corpus is a large and structured set of texts. The corpus-based method of information processing gives language researchers advantages to not to rely on native speakers' or even on their own perception or even on dreamt up examples. There are incessant research efforts to practice corpus-based methods as the leading part in any language processing systems. Since our grammar checker is data-driven, therefore, to make our grammar checker vastly accurate and robust, we had to build the Bangla monolingual corpus named NHMono01. Later, we have extracted the corpus to construct the Bangla monolingual lexicon to enhance the competence of the spell checker. Now, according to our knowledge, NHMono01 is the largest Bangla Monolingual corpus with over 100 Million Bangla words. In addition, our lexicon is the largest Bangla monolingual lexicon with over 1 Million Bangla words.

2.2 Limitation of Previous Approaches

The necessity of building our own monolingual corpus triggered due to the lack of notable monolingual corpus in Bangla. Our study shows only two Bangla monolingual corpora are notable. One is BDNC01 [6] and another is SUMono [7]. However, both the corpora have major limitations on the following:

- I. Small Corpus Size
- II. Corpus Creation Time

In 2014, Md. Abdullah Al Mumin et al. proposed a Bangla Monolingual Corpus SUMono which contains 27 million words collected from various sources [7]. This is currently the largest Bangla monolingual corpus. Md. Farukuzzaman Khan et al. proposed another Bangla monolingual corpus BDNC01 in 2017 which contains only 12 million words. The authors proposed the same monolingual corpus previously with only 11 million words in

2012 [8]. However, the size of both corpora is not enough to build a robust corpus-based system. On the other hand, NHMono01 consists of over 100 Million Bangla words collected from various sources.

Another limitation is that both the corpora took huge time to build. According to the authors of BDNC01, it took 6 years to build the corpus [6] and it took about 4 years to build SUMono [7]. For the reason that, all these corpora created manually by typing, copying and pasting [6]. No wonder why these corpora took so much time to be built. On the other hand, our NHMono01 corpus was created within 2 months. The reason for this unbelievable timeframe is that instead of manually copying and pasting, we have developed spider bot to crawl and scrape text using the latest python framework to collect text automatically in the desired way. Section 2.3 describes how we have built the spider bot and how we have collected data from various sources described in domains section 2.4. In addition, Mithun Biswas et al. in 2017 demonstrated a handwritten Bangla character dataset comprises 166105 character images for the better OCR output [23]. However, their work is limited to individual characters only and handwritten. Therefore, we don't compare our corpus with their dataset.

2.3 Creating Web Crawler and Scraper

Using a spider bot to crawl and scrape text from any website is the stress-free way to accumulate information or text. We have developed spider bots to crawl and scrape diverse websites automatically. In order to build a spider bot we have used Python's Scrapy framework. Scrapy is an open source framework for extracting data from any website. We have adopted JetBrains's PyCharm as an IDE to code the spider. The Scrapy framework requires installing several environments and other necessary frameworks. The commands and other minutiae could be found in the authorized website of Scrapy framework [9]. We have described briefly the required actions to create, crawl, scrape and store data using the spider we have developed.

- i. First, we make sure we have downloaded and installed the latest version of Python. Details can be found in [10].
- ii. Install PyCharm IDE, downloaded community version. It is free and open source.

```
pip install scrapy
```

iii. Install Scrapy using following command in command prompt,

vi. After successfully install scrapy framework. We are ready to create our spider.

v. Open a scrapy project using the following command,

```
scrapy startproject NHMono01_project
```

This command creates a folder named NHMono01_project with some obligatory .py files in it.

vi. Inside a project, we can create multiple spiders for manifold websites. The same spider can also be used for numerous websites just need negligible changes in the program. Below, we have demonstrated with an example of how we have developed a spider to scrape “Prothom Alo” which is the most widespread online newspaper in Bangladesh.

```
scrapy genspider SpiderProthomAlo www.prothomalo.com
```

This command created a .py file with the spider named SpiderProthomAlo. The spider contained some basic program coded in it.

vii. Now, this is the place where we require modifying the code according to our necessity. Since this illustration is for Prothom alo, hence, the basic program was designed to scrape www.prothomalo.com automatically by the spider. The code snippet illustrates the core segment of the spider created to scrape the whole Prothom Alo online website.

Code Snippet 1: Crawling and Scraping Prothomalo.com

```
import scrapy

class SpiderProthomAloSpider(scrapy.Spider):
    name = 'SpiderProthomAlo'
    allowed_domains = ['www.prothomalo.com']
    start_urls = ['http://www.prothomalo.com/archive/2018-01-24']
```

```

def parse(self, response):
    item_link = response.css('a.link_overlay::attr(href)').extract()
    file_object = open(".txt", "a", encoding='utf-8-sig')
    for i in range(0, len(item_link)):
        item_link_text = response.css('p::text').extract()
        file_object.write(item_link_text[i] + "\n")
        next_page = item_link[i]
        if next_page is not None:
            next_page = response.urljoin(next_page)
            yield scrapy.Request(next_page, callback=self.parse)

```

Since this spider was created to scrape Prothom Alo, in the `allowed_domains` variable we put the URL of the website of Prothom Alo. Then, we mentioned the URL to start scraping in `start_urls` variable. In this case, we started our scraping from the date 2018-01-24 and the spider automatically crawled all the articles available from the beginning of Prothom Alo's online publication to this date. `item_link` variable contains all the URLs needed to crawl and we specified where to find those URLs in Prothom Alo's website. We appended a text file in UTF-8 mode named `scraped_data_prothomalo` to store all the texts in it. Inside a loop, we had extracted all the text contents contained in paragraphs from each page and wrote it to the text file. This spider will automatically crawl, scraped and stored all the articles in Prothom Alo's online version.

Another limitation is that both the corpora took huge time to build. According to the authors of BDNC01, it took 6 years to build the corpus [6] and it took about 4 years to build SUMono [7]. For the reason that, all these corpora created manually by typing, copying and pasting [6]. No wonder why these corpora took so much time to be built. On the other hand, our NHMono01 corpus was created within 2 months. The reason for this unbelievable timeframe is that instead of manually copying and pasting, we have developed spider bot to crawl and scrape text using the latest python framework to collect text automatically in the desired way. Section 2.3 describes how we have built the spider bot and how we have collected data from various sources described in domains section 2.4.

2.4 Corpus Creation

In order to create a structured and balanced corpus, we have collected text from various domains such as newspaper, computer science and IT, literature, personal blogs, and textbooks. Although we have tried to keep the corpus as balanced as possible, most of our texts collected from several reliable and popular online newspapers and blogs. We have used the following machine to run the spider and collect text.

Table 1: Abstract configuration of the device used in scraping

Name:	Inspur NF5170M4 1U Server
Processor:	Intel®Xeon®E5-2600V3 or V4 Processors
Chipset:	Intel ®C610 server chipset
Memory Used:	32GB
Storage Used:	2TB

Full specification of the machine can be found in [11]. The following actions we did to collect the data and create the corpus.

i. First, we have decided from which domains we would have to collect data. We have decided to collect data from the following domains.

- a) Newspapers.
- b) Computer Science and IT.
- c) Literature.
- d) Personal Blogs
- e) Textbooks.

ii. In order to collect data from the above sources we had decided to crawl and scrape the following places.

- a) Popular Bangla online newspapers (www.prothomalo.com , www.banglanews24.com , www.kalerkantho.com, bangla.bdnews24.com, etc.)

b) Popular Bangla Computer Science and IT websites (www.techtunes.com.bd , www.cnewsvoice.com etc.)

c) Popular Bangla online blogs (www.somewhereinblog.net, www.sachalayatan.com etc.)

d) Bangla textbook (www.nctb.gov.bd), we have done this portion manually. We have manually downloaded the PDF version of the textbook and later converted it to plain text.

iii. After deciding which website to scrape and which portion of the website to scrape we modify the spider's program accordingly. We spend a few hours analyzing each website since we have seen the structure of each website if differ from another. Moreover, we needed to avoid all the other things except pure Bangla Unicode text in an article (like text in titles, menus, ads, pictures, English sentences etc.). Thus, for each website, we changed the program of the spider accordingly so that the output text file contains pure Bangla Unicode only.

iv. And then, we ran the spider using the following command (the following example is for the spider we have shown in section 2.3).

```
scrapy crawl SpiderProthomAlo
```

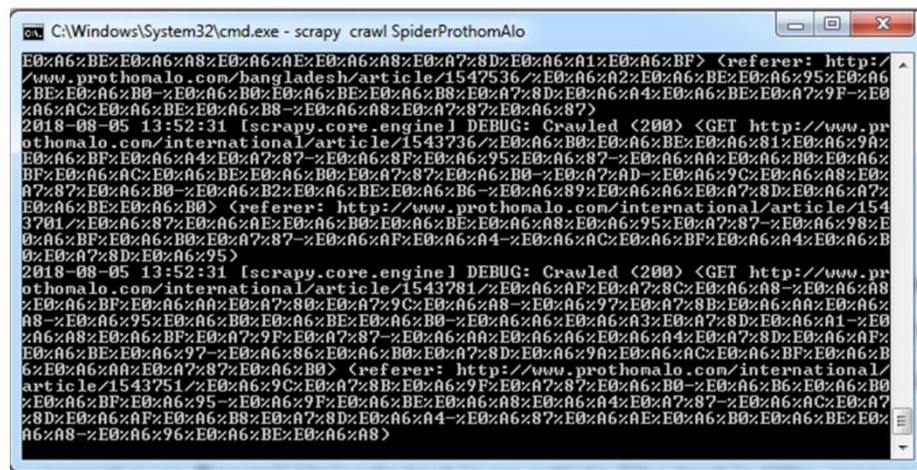


Figure 1 : Our spider crawling and scraping on prothomalo.com

v. Sometimes we did trial runs of the spider to test whether our spider fetching only pure Bangla Unicode text or not.

vi. Execution time: According to our study, we have seen each website took a minimum of 5 hours and a maximum of 2 days to be completely scrapped and stored.

vii. After a successful scraping operation, we have got a text file that contains all the pure Bangla Unicode text of a particular website.

viii. We spent a few hours checking everything is ok in the text file. If no imperfection is found, then we store the texts in our central text file.

ix. We spent slightly more than 2 months to create the complete corpus by doing the above-mentioned actions repeatedly.

Our corpus consists of over 100 million Bangla words. This massive Bangla monolingual corpus has been created just in 2 months. This remarkable timeframe achieved due to the use of modern tactics to collect data. Table 2 shows a comparison among all Bangla monolingual text corpuses.

Table 2 : Comparison all Bangla Monolingual text corpus

Corpus Name	Corpus size	Creation time	Technique of data collection
BDNC01 [6]	12 Million Words	6 Years	Manual- Copy and Paste
SUMono [7]	27 Million Words	4 Years	Manual - Copy and Paste
NHMono01 (Our Corpus)	Over 100 Million Words	2 Months	Automatic – Crawling and scraping with powerful Python’s Scrapy framework.

2.5 Lexicon Creation

A lexicon is an inventory of lexemes. Lexicon also contains bound morphemes, compound words and idiomatic expressions. On the other hand, a typical dictionary contains only root words usually excluding bound morphemes [12]. Lexicon is a core part of a spell checker. Our research shows no notable Bangla monolingual lexicon available in Bangla. Therefore, we have created our own lexicon by extracting words from our monolingual text corpus NHMono01. Creation of the lexicon was easy since we already had the corpus. According to knowledge, this lexicon is the largest monolingual text lexicon in Bangla with over 1 million words. Following the actions required to create our lexicon. The language used during the period of creating the lexicon is Pure Python and used the same machine describes in subsection 2.4 in Table 1.

- i. Create a copy of the corpus file; this copied corpus will be converted to the lexicon.
- ii. Read the copied corpus file.
- iii. Remove unnecessary punctuations such as, ‘.’ (full stop) , ‘,’ (comma) etc.
- iv. Remove any foreign words and characters such as an English word.
- v. Remove any special characters like @, #, \$, ©.
- vi. Remove redundancy of words from the corpus file. For the reason that, in the corpus, we have the same word in multiple places. However, we need only unique words.
- vii. Take each word and move it to a new file and sort the file in alphabetical order (words starting with the character ‘অ’ (0985) should be at the beginning and ‘ঝ’ (09B9) at the last). Now we have a lexicon with unique words and without garbage words or characters.
- viii. Finally, the lexicon goes through a final filter. If we look at the following Bangla sentences,

a) উত্তরে রিক্সা চালক বললেন “আইজকা জামু না”।

b) আগামীকাল রোববার ঢাকায় আসছেন বিশ্বব্যাংকের প্রেসিডেন্ট জিম ইয়ং কিম।

We can see some unwanted and uncommon Bangla words (আইজকা, জামু, জিম, ইয়ং, কিম). We have to remove these unwanted words from our lexicon. However, in the beginning, it seems difficult for us to distinguish these unwanted words from general words. Finally, our study finds the best way to remove these unwanted words to find the frequency of each word in our corpus NHMono01. The following algorithm describes the procedure of solving the issue.

Algorithm 1: Filtering lexicon file

1. Lex $\leftarrow$ read the whole lexicon file.
2. LOOP of the size of Lex
3. wordLex $\leftarrow$ Take a word from Lex
4. wordFreq $\leftarrow$ find word frequency of wordLex in CorpusNHMono01
5. IF wordFreq < 5

```

6.         remove the word wordLex from Lex immediately
7.     ELSE IF wordFreq >= 5 and wordFreq <=10
8.         temp_store_array ← wordLex
9.     ELSE
10.        // do nothing. wordLex is a regular frequent word
11.    END IF
12. END LOOP
13. check each word from temp_store_array manually. If any word seems
    unfit remove it from temp_store_array
14. combine Lex and temp_store_array
15. write it to the final lexicon file

```

Our study shows, according to our NHMono01 corpus if a word frequency is less than 5 than it is usually unwanted and garbage word. Therefore, we remove it immediately. However, if the word frequency is in between 5 and 10, we cannot guarantee that the word is a garbage word. Thus, we check those words manually and only correct words are combined with the final lexicon file.

This is how we have developed our own lexicon from our own corpus NHMono01. The lexicon consists of over a million unique Bangla words, including free and bound morphemes. This lexicon has used to enrich our spell checker.

2.6 Distributing Lexicon with Pre-defined Double Metaphone Values

We have distributed the lexicon into the smaller lexicon so that we have to process very few words instead of the whole lexicon. This boosts the processing speed and reduces the time. We have distributed the lexicon using a very primitive hash function which returns only the lexeme size. We have distributed the lexicon using their string length. With each lexeme we also store its double Metaphone values. This is because we have tried to find the double Metaphone values of lexemes on the fly. However, this slowed down our system. Using pre-defined double Metaphone values significantly reduced the processing time of our proposed system.

Algorithm 2: Distributing lexicon with pre-defined Double Metaphone values

1. MLex ← Input the whole Bangla monolingual lexicon
2. Li ← Split MLex into lexemes using whitespaces
3. Ki ← generate hash key from each lexeme Li (simple string length)

4. $DMV \leftarrow$ find double metaphone values of lexeme Li using our double metaphone algo.
5. Insert lexeme Li along with its DMV into distributed lexicon- Ki

The distribution of lexicon with pre-defined double metaphone process is a single time process. We do not need to do this anymore unless we want to update or upgrade our system in the future. The algorithm below demonstrates the procedure of distributing lexicon with pre-defined double metaphone values.

2.7 Sampling, Biasness and Strength

The size of our corpus NHMono01 is enormous and it is a balanced corpus. This means that the texts are collected and compiled according to specific principles, such as different genres, categories or styles of Bangla language. As mentioned in section 2.4 to make our corpus balanced we have collected text of different categories from different sources. We have followed the Quota Sampling technique which is a non-probability based sampling technique [24]. The reason behind the quota sampling is that the data available on the internet is enormous which makes building a balanced corpus quite difficult. Therefore, the only option left to us is to select the appropriate samples with the same amount of data for all categories. However, selecting the appropriate sources of texts was one of the difficult tasks.

In order to reduce biasness in data, we have spent ample time analyzing different sources whether they are appropriate or not. However, there may have some degree of inaccuracies in the sources of the texts.

Although our corpus might contain negligible biased data, the strength of it is that it has enormous data of different categories from various sources which make the corpus balanced, well-organized and less biased. In addition, as section 2.4 and 2.5 describe, we have reduced the biasness by automatically and manually filtering garbage data. Moreover, the result of a statistical approach mostly depends on the amount of data. Therefore, a small amount of biasness does not necessarily affect the system in all the cases.

Chapter 3

Spell Checker

3.1 Introduction

Checking for any kinds of spelling mistake is very essential to ensure good content accessibility and readability for the readers. Moreover, an online article free from spelling mistakes gives better crawl-ability and indexing for search engines. This work can be easily done by a spell checker. Detecting erroneous word is not that grim for a spell checker. However, suggesting the appropriate words for an erroneous word is the most challenging part of a spell checker. As we have already mentioned in section 1.1 that a very few researches have done on Bangla spell checker even though thousands of people are writing books, article, blogs in Bangla every day in Bangladesh and West Bengal of India. Currently, available spell checkers in Bangla is not at all fruitful in various aspects. Our study shows, these approaches have several limitations on real-life text processing. By observing this necessity we have proposed a spell checker combined with a grammar checker by mentioning all the limitations and their solution. The description of the grammar checker can be found in Chapter 4. Previous works of spell checker are mentioned in section 1.2. Subsection 3.1.1 to 3.1.5 describes some basic terminology relevant to our spell checker.

3.1.1 Soundex

Soundex is a phonetic algorithm designed in 1900's. Soundex algorithm phonetically encodes group similar sounding consonant characters. The process usually excludes vowels except the vowels at the beginning of the word. You can find Kaykobad's Bangla soundex encoding table in [3,19] and Mumit's Bangla soundex table in [4]. According to the Mumit's soundex encoding algorithm returns the following output.

Table 3 : Sample result of N. UzZaman and Mumit's Bangla soundex table

Input	Encoded	Suggestion
খুমাড়	kumar	কুমার
পাসান	pasan	পাশাণ

Soundex is an ancient version of phonetic algorithm and has lots of imperfections. Therefore, we have not used soundex in our spell checker.

3.1.2 Metaphone

Metaphone is another phonetic algorithm developed in 1990 by Lawrence Philips [14]. Metaphone algorithm improved earlier soundex algorithm in several aspects. Metaphone algorithm is more intricate than previous algorithms because it included new rules to handle different spelling inconsistencies.

3.1.3 Double Metaphone

In the year 2000, the same author of Metaphone proposed a newer version of the Metaphone called Double Metaphone [14]. This is a highly sophisticated phonetic algorithm containing all multifaceted rules to grip several types of elocution. In addition, it makes name searching simpler since double metaphone considers pronunciation of diverse names as well.

Double metaphone provides a primary and a secondary code for a single word that is why it's called "Double". In our spell checker, we have implemented Double Metaphone phonetic algorithm, though we tried to implement the latest version Metaphone 3. In subsection 3.1.4 we have explained why we could not use Metaphone 3.

3.1.4 Metaphone 3

Lawrence Philips later released the final version of metaphone algorithm in 2009 called Metaphone 3. The author claimed that he improved the accuracy 89% of Double Metaphone to 98% [14]. This is the most advanced version of the phonetic algorithm. However, it is a commercial product sold as source code by the author. At the very beginning of our study, we planned to implement Metaphone 3 as our phonetic algorithm. Unfortunately, we failed due to the fact that the author released it for commercial purposes [14]; hence he did not publish any detailed algorithm of Metaphone 3.

3.1.5 Edit Distance

Edit distance is used to determine dissimilarities between two strings or words by calculating the minimum number of actions required to transform one string into the other [13]. We have used Levenshtein's edit distance algorithm to determine the difference between two words. The Levenshtein distance between two strings a , b is specified by $lev_{a,b}(|a|,|b|)$ where, (assuming the length of a , b are $|a|$ and $|b|$ respectively)

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

Equation 1: Levenshtein's Edit Distance Formula [13]

The table below shows the edit distance between the misspelled Bangla word “বিদান” and some words from the lexicon.

Table 4 : Levenshtein's distance between two words.

Lexicon word	Levenshtein's distance with misspelled word “বিদান”
দান	2
বিধান	1
অনুদান	3

From the above table, we can see smallest edit distance with word “বিদান” is the word “বিধান”. The Levenshtein's distance can be used to find the dissimilarities between two double metaphone encoded strings. For example, our double metaphone encoding technique returns “bdn” for the misspelled word “বিদান”. Table 5 illustrates the Levenshtein's distance between two double metaphone encoded words.

Table 5 : Levenshtein's distance between to double metaphone encoded words.

Lexicon word	Double metaphone of lexicon word	Levenshtein's distance with misspelled double metaphone encoded word “bdn”
দান	dn	1
বিধান	bdn	0
অনুদান	ndn	1

Table 5 illustrates the double metaphone encoding distance between word “বিদান” and “বিধান” is 0. Hence, our system assumes the correct word for the misspelt word “বিদান” should be “বিধান”.

3.2 Types of Spelling Errors

According to Damerau[15] all spelling mistakes can be divided into two types.

3.2.1 Typographic Error

Typographic errors occur due to typing mistakes. We usually place our finger on the wrong key while typing faster on the keyboard or tapping on a screen. This type of error usually does not fall under any linguistic rules. Example of Typographic error is given below,

Table 6 : Typographic error example in Bangla

Error Type	Misspelled Word	Correct Word	Transliteration in English
Insertion	দেইশ	দেশ	Desh (translation: Country)
Deletion	বাংল	বাংলা	Bangla
Substitution	বাংলাদেশ	বাংলাদেশ	Bangladesh
Transposition	কমল	কলম	Kolom (translation: Pen)

3.2.2 Cognitive Error

Cognitive error is also called Phonetic error. Cognitive error occurs due to the similar pronunciation of alphabets. For example, in Bangla, alphabets ‘প’(PA) and ‘ফ’(PHA) sounds the same. In Bangla, there are so many other alphabets which sound same (‘গ’(GA)-‘ঘ’(GHA), ‘শ’(SHA)-‘স’(SA)-‘ষ’(SSA), ‘ব’(BA)-‘ভ’(BHA) etc.). That is why the cognitive error is more frequent in Bangla. Some example of cognitive errors is illustrated below,

Table 7 : Cognitive error example in Bangla

Misspelled Word	Correct Word	Transliteration in English
দেস or দেশ	দেশ	Desh (translation: Country)
বাষা or ভাষা	ভাষা	Bhasha (translation: Language)
গোলাফ or ঘোলাপ	গোলাপ	Golap (translation: Rose)

On the other hand, Zeeshan Bhatti [16] proposed two more error types for the Sindhi language in 2012. They are the followings,

3.2.3 Visual Error

According to the author, a visual error occurs due to the visually same shape of alphabets in Sindhi. The author also called it “Similar shape errors” [16]. Our study shows in Bangla we have some similar looking alphabets as well. Table 6 shows the alphabets which look similar in Bangla.

Table 8 : Almost similar Bangla alphabets

Similar Alphabets in Bangla
উ (U) & উ (UU)
ব (BA) & র (RA)
ন (NA) & ণ (NNA)
ড (DDA) & ড় (RRA)
ঢ (DDHA) & ঢ় (RRHA)
য (YA) & য় (YYA)
ষ (YA) & ষ (SSA)

Table 9 : Visual error example in Bangla with International Phonetic Alphabet(IPA)

Misspelled Word	Correct Word	Translation in English
পাহাড (IPA: paɦaɖ)	পাহাড়(IPA: paɦaɽ)	Mountain
বিশেষ(IPA: biʃeɖʒ)	বিশেষ(IPA: biʃeɽ)	Special
ফুটরল(IPA: pʰuɽɾɔl)	ফুটবল(IPA: pʰuɽbɔl)	Football

These similar alphabets may lead to critical visual errors illustrated in Table 9.

3.2.4 Space Related Word

Adding an unnecessary space inside a word or removing a necessary space between two words is a space related error. Space related errors occur frequently in Bangla text unintentionally. According to our study, in Bangla, we have so many words people unnecessarily put space between two word fragments and in some places we saw the opposite. Table 10 shows some example of space related errors.

Table 10 : Space related errors example in Bangla

Type	Misspelled Word	Correct Word	Transliteration in English
Insertion	মোটর গাড়ি	মোটরগাড়ি	Motorgari (Translation: Automobile)
Deletion	মিষ্টিআলু	মিষ্টি আলু	Mishti Alu(Translation: Sweet Potat)

Although, in the above-mentioned space related error example, our spell checker will not detect “মোটর গাড়ি” as a misspelt word due to the fact that those word fragments are individually meaningful in Bangla and can stand alone. However, this particular space related error can be detected only by considering the context of the word in a sentence.

3.3 Limitation of Previous Approaches and Solution

The limitations of previous spell checkers are explained in this section. Grammar checker section can be found in section 4.2. As we mentioned earlier, very few researches have been done on this section of natural language processing in Bangla language. Several limitations observed on the previous approaches during our study on the Bangla spell checker. As we have mentioned earlier, most of the limitations observed when processing real-life text. In addition, these approaches did not mention the features of spell checker which cannot be ignored. Most common and mentionable boundaries are as follows.

3.3.1 Trivial Lexicon size

In order to create a vigorous spell checker, it must have a sufficiently large lexicon. However, all the previous approaches used a very trivial lexicon. In addition, most of these researches not clearly mentioned their lexicon, lexicon size and how they created lexicons. In the last research of N. UzZaman and Mumit Khan mentioned they have used only 1607 misspelt word [14] to test their spell checker and ignored to mention about their lexicon.

By observing the necessity of large lexicon in order to build a robust Bangla spell checker we have built an immense Bangla lexicon which consists over a million unique Bangla words including almost all free and bound morphemes. The building procedure of the lexicon and other details can be found in Chapter 2.

3.3.2 Primitive Approaches

In all the previous approaches, we have observed to use very primitive phonetic algorithms, such as Phonix and Soundex in [3,4,19]. B.B Chowdhury[2] used his own phonetic algorithm similar to Soundex algorithm. His reverse order dictionary based system needs more memory and processing time. Although in 2005, N. UzZaman and Mumit Khan [5] projected and applied advanced Double Metaphone algorithm, it has several limitations we have fixed.

As a solution, we have applied the latest phonetic algorithm Double Metaphone. Although in October 2009, Lawrence Philips, the same author of Double Metaphone released the most advanced version of metaphone encoding called Metaphone 3, it is a commercial product sold as source code. Due to lack of algorithm, we could not use Metaphone 3. Moreover, we have used the latest techniques, e.g. hashing and pre-defined/pre-calculated data in every step which helps us to speed up the processing time and increased the efficiency of the spell checker.

3.3.3 No Hashing

All the previous approaches have used, negligible data in their approaches. Therefore, they did not consider using hashing in their spell checker. However, for a fruitful spell checker, large amount of data is essential. To handle a large amount of data, to increase efficiency and reduce processing time we need hashing techniques.

When a dataset is small, we don't really worry about the processing time. However, processing speed increase dramatically with the increased amount of data. We have used hashing techniques to distribute the lexicon into several smaller lexicons. When we process a single lexeme we find the key by hashing the lexeme and directly search for that word the specific lexicon. We don't need to process all the other lexicons. This reduces the processing time dramatically. Whenever we need any data read, write, delete or update we have used our hashed function to do that to speed up our process.

3.3.4. Not Suitable For Big Data Processing

Data processing time increases simultaneously with the increasing data. Since all the previous approaches used a very small dataset (mentioned in subsection 3.3.1), they did not consider techniques to reduce processing time. However, there should have applied techniques such as hashing or pre-defined/pre-calculated data which could boost up data processing time.

As mentioned in 3.3.3, in order to reduce data processing time we have used hashing techniques. However, hashing alone is not enough to process a big amount of data like our monolingual corpus or the lexicon. We have observed that calculating DMV of lexemes (on the fly) takes long processing time. Since for a long article we have to calculate many Double metaphone values, the system will miserably slow. Thus, we have decided to use pre-defined/calculated DMV and we find this makes the system much faster than before.

3.3.5 Handling Foreign Words and Numbers

We often see English words in Bangla article. Moreover, there are several occurrences of Bangla numbers in the sentences. All the previous approaches handle neither foreign words nor numbers.

As a solution, our system process both foreign words and Bangla numbers. If our approach detects a foreign word using Unicode of characters, then it just avoids processing it and marking the lexeme with blue color. Our approach doesn't decide whether the spelling of a foreign word is correct or not. If our approach detects a lexeme has started with a number (using Unicode) then it checks if the lexeme is completely numerical or alphanumerical. If the lexeme is completely numerical, then there is nothing to check. Mark the lexeme as correct with green color. If the lexeme is alphanumerical, then checks the alpha-part has one of the following such as 'ম, ঝ, থ, ভম, টি, টা' etc. If it matches the aforementioned criteria, then our approach mark it as correct with green color, otherwise incorrect with red color.

3.3.6 No Association of Spell Checker with Grammar Checker

According to our study this is the most important limitation we found in previous approaches. In all the previous approaches, they have done researches either on spell checker or on a grammar checker. There is no such approach that combines both spell and grammar checker. However, we know, to write a fruitful Bangla article we need both the spell and grammar checker.

The aforementioned problem gave us the motivation to build both the spell and grammar checker combine. Thus, we have combined both spell and grammar checker to provide a complete error check in a Bangla article explained throughout this thesis.

3.3.7 No Customized User Experience

Displaying results based on a user's previous activities or selection can help a user to decide faster and make an article spell free faster. However, we have seen in any previous approaches no customized user experience feature available.

In order to provide a customized view to a user, our approach store decisions and choices of a user in a local log. This local log used for future suggestions. For example, if a user writes WL instead of RL, we suggest a list of words including RL as a suggestion of the incorrect word WL. However, RL could be anywhere in the suggestion list. When the user clicks on a word, for example, RL, then our approach store RL, WL and frequency of selecting RL for WL. Next time, whenever the user mistakenly writes WL again, we show the most frequently chosen suggestion word at the top of the suggestion list for the word WL. In this example, RL will be at the top of the suggestion list. This gives a user a fast and efficient way to choose the correct word from the suggestion list. We have used the same way in our grammar checker to store grammar suggestions.

There are other negligible limitations observed in the previous approaches. Most of the previous approaches neither solve these aforementioned issues nor mentioned or clarified these issues in their study. However, we have tried to solve each of these issues.

3.4 Our Double Metaphone Encoding

We have used Double Metaphone encoding as our phonetic algorithm to detect phonetically similar Bangla words. Double metaphone provides a primary and a secondary code for a single word which helps to identify different pronunciation of the same alphabet. Our encoding table is based on the encoding table projected by N. UzZaman and M. Khan's (can be found in [5]) which is the only double metaphone available in Bangla. Our study shows this encoding table is the most accurate encoding for Bangla, thus we have used the encoding table with some improvements. Such as,

i. Letter 'ঋ'(U+098B) coded 'r' instead of 'ri':

We coded the letter 'ঋ' as 'r' instead of 'ri' in our encoding table.

Table 11 : Metaphone values of 'ঋ' (VOCALIC R)

	Letter	Name	Unicode	Code
Previous Approach[5]	ঋ	VOCALIC R	U+098B	' ri '
Our Approach				' r '

This is because, for a proper suggestion edit distance should be as less as possible where 0 considered the closest. If compare both codes ('ri' and 'r') side by side we will observe the difference.

Table 12: Example comparison of 'ঋ' between previous and our approach

'ঋ' coded 'ri' (Previous Approach [5])			'ঋ' coded 'r' (Our Approach)		
Word	Primary Code	Edit Distance	Word	Primary Code	Edit Distance
ঋষি)correct(	Ris	ED(ris rs) = 1	ঋষি)correct(	rs	ED(rs rs) = 0
রিশি)incorrect(	Rs		রিশি)incorrect(	rs	

Table 12 demonstrates an example code 'ri' results edit distance of 1. On the other hand, code 'r' in our approach results edit distance of 0. Both 0 and 1 give 'ঋষি' as a suggestion of incorrect word 'রিশি'. However, displaying appropriate suggestion at the top of suggestion list is one of the major tasks in a spell checker. In our approach, the word 'ঋষি' will be suggested at the top of the suggestion list wherein the previous approach will not.

ii. Letter ‘ꠘ’(U+09C3) coded with ‘r’:

Table 13 : Metaphone values of ‘ꠘ’ (Vowel Sign Vocalic R)

	Letter	Name	Unicode	Code
Previous Approach [5]	ꠘ	Vowel Sign	U+09C3	Not mentioned
Our Approach		Vocalic R		‘ r ’

We have observed, there is no mention of the letter ‘ꠘ’ in [5] separately. The Authors of that paper gave an example which demonstrates they considered the letter ‘ꠘ’ same as ‘ꠕ’. However, these two letters are different with different Unicode U+098B and U+09C3 respectively. Table 14 shows an example of this scenario.

Table 14 : Comparison of coding ‘ꠘ’ between previous and our approach

‘ꠘ’ not coded(Previous Approach [5]))letter ‘ꠘ’ has different Unicode U+09C3, U+098B and U+09C3 are not the same)			‘ꠘ’ coded with ‘r’ (Our Approach)		
Word	Primary Code	Edit Distance	Word	Primary Code	Edit Distance
বিকৃত) correct(	bkt	ED(bkt bkrt) = 1	বিকৃত) correct(	bkrt	ED(bkrt bkrt) = 0
বিকরিত)incorrect(	bkrt		বিকরিত)incorrect(	bkrt	

In Table 14 we have demonstrated the consequences of not coding the letter ‘ꠘ’ individually. No coding the letter provides edit distance of 1, whereas in our approach (coding the letter with 'r') provides edit distance of 0. This makes the suggestion more accurate and efficient in our approach.

iii. Unicode correction for letter ‘ꠗ’:

In the previous approach [5], we have observed incorrect mapping for letter ‘ꠗ’(Rha) to the Unicode ‘U+09A2’ which is actually the Unicode of letter ‘ꠗ’(Ddha). Although both the letters look almost the same, pronunciation and meaning of these two letters are very different. Therefore, we have changed the mapping to the correct Unicode.

Table 15 : Unicode and name correction for letter ‘र’

	Letter	Name	Unicode	Code
Previous Approach [5]	र	DDHA	U+09A2	‘ r ‘
Our Approach		RHA	U+09DD	

We have also made few negligible changes and furnished the encoding table of previous approach before it has been used in our proposed approach.

3.5 Algorithm and System Diagram of Spell Checker Section

The system diagram of our proposed spell checker section has been given below. The system diagram of other section (grammar checker) can be found in Chapter 4.

Table 16 : Table of terminology (spell checker)

Term	Definition
DMV	Double Metaphone Values
DMA	Double Metaphone Algorithm
UED	Unicode-based Edit Distance
ED	Edit Distance
SD	Sorted Descendingly

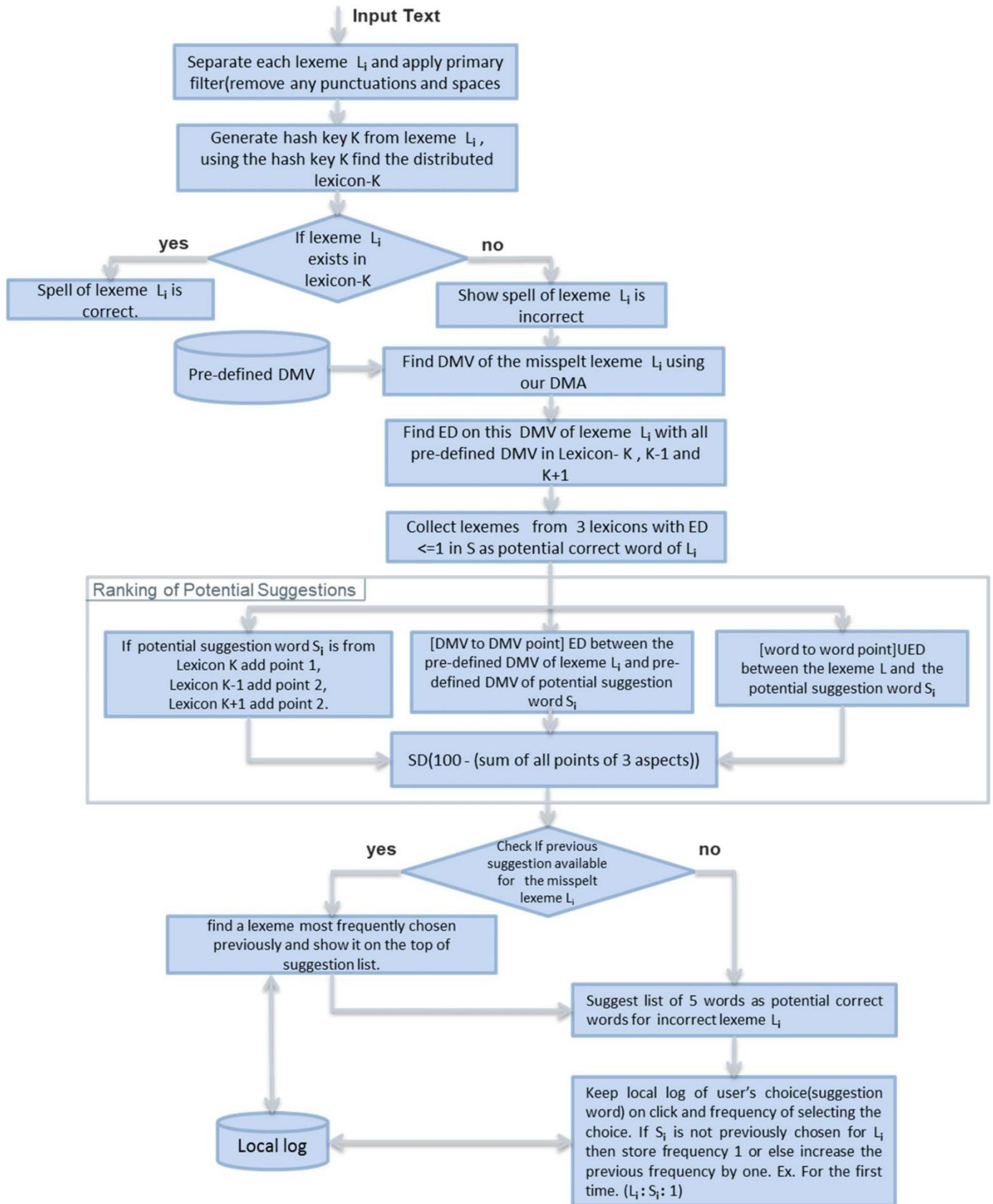


Diagram 1 : System diagram of spell checker section

The algorithm of spell checker is given below.

Algorithm 3 Spell Checker

```

1. IN  $\leftarrow$  input text
2.  $L_i \leftarrow$  split IN into separate lexeme using whitespace
3.  $L_i \leftarrow$  remove punctuations and whitespace(s) of  $L_i$  (will be added back later after processing)
4.  $K \leftarrow$  generate hash key from  $L_i$ 
5. IF  $L_i$  exists in lexicon K
6.   mark  $L_i$  as Correct lexeme
7. ELSE
8.   mark  $L_i$  as Incorrect or misspelt lexeme
9.    $DMV \leftarrow$  generate double metaphone values(DMV)
10.  calculate levenshtein's edit distance(ED) of DMV with all pre-defined DMV of lexemes in lexicon K, K-1 and K+1
11.   $SUG \leftarrow$  store all the lexemes where  $ED \leq 1$ 
12.  FOR EACH in SUG
13.    (Generate rank of  $SUG_k$  for better suggestion)
14.     $SUM \leftarrow 0$ 
15.    IF  $SUG_k$  belongs to lexicon-K
16.       $SUM \leftarrow SUM + 1$ 
17.    ELSE ( for lexicon K-1 and K+1 )
18.       $SUM \leftarrow SUM + 2$ 
19.    END IF
20.     $EDDMV \leftarrow$  generate ED between DMV of  $L_i$  and DMV of  $SUG_k$ 
21.     $SUM \leftarrow SUM + EDDMV$ 
22.     $EDWord \leftarrow$  generate ED between  $L_i$  and  $SUG_k$  ( text to text)
23.     $SUM \leftarrow SUM + EDWord$ 
24.     $RANKSUG_k \leftarrow 100 - SUM$ 
25.     $RANK \leftarrow$  Store  $SUG_k$  and  $RANKSUG_k$ 
26.  END FOR
27. Create Suggestion List SLIST
28. IF previous suggestion available for  $L_i$  in local log
29.    $MFU \leftarrow$  Find most frequently chosen correct word for lexeme  $L_i$ 
30.   Insert MFU at the top position of SLIST
31. END IF
32. add 5 Top Ranked words to SLIST from RANK
33. present all lexeme in SLIST as tentative suggestion for misspelt lexeme  $L_i$ 
34. keep local log of user's choice for future suggestion with misspelt word : correct word from SLIST : total frequency of this such as  $L_i : SLIST_i : 1$  for the first time.
35. END IF

```

The algorithm above demonstrates how our grammar checker works from taking input text to providing results and giving appropriate suggestions for incorrect words. Algorithm of grammar checker section can be found in Chapter 4.

Chapter 4

Grammar Checker

4.1 Introduction

This chapter demonstrates our grammar checker system. Checking for grammatical mistakes is also very essential as checking for spelling mistakes. The combination of both spell checker and grammar checker is very essential for a successful Bangla article. As we have already mentioned in subsection 1.1 that a very few researches have done on Bangla spell and grammar checker. In addition, there is negligible notable researches have been done on Bangla grammar checker. Currently, available grammar checkers in Bangla is not at all prolific due to the very trivial dataset and primitive way of processing data which led the system to be responsible for the very low accuracy. Previous works of grammar checker are mentioned in section 1.2.

4.2 Limitation of Previous Approaches

In this section, we have mentioned some important limitations of previous grammar checkers. Several limitations observed on the previous approaches during our study on Bangla grammar checker. Most common and mentionable boundaries are followed,

4.2.1 Minor Corpus

In order to create an ample grammar checker, it must have a sufficiently large corpus which consists of different types of sentences. However, all available grammar checking approaches used a very trivial corpus. According to our research at present only one notable work is done on Bangla grammar checker which is N. UzZaman and Mumit Khan's n-gram based grammar checker. However, their dataset consists only 5000 words collected from Prothom-alo newspaper [17]. In 2012, Hasan, K. M. et. al. proposed a CFG which can be used in Bangla grammar checker which used only 225 sentences to figure out their success rate [18].

The above-mentioned data sets are not even comparable with our corpus size. By observing the necessity of large corpus in order to build a robust Bangla grammar checker we have built an immense Bangla corpus which consists over 100 million unique Bangla words (over 130 thousands article). The building procedure of the corpus and other details can be found in Chapter 2. The comparison with other corpus has been given in section 2.4.

4.2.2 Low Accuracy

Low accuracy of a system make a system inefficient and force us to avoid using the system. Our study shows no previous Bangla grammar check provided minimal accuracy rate required for an efficient grammar check. N. UzZaman and Mumit Khan's grammar checker has an accuracy rate of 53.7% only [17].

Since language model systems are based on the dataset. Therefore, in the n-gram system, more data assured more accuracy rate. Since we have a large corpus our system is capable of finding complex errors and suggesting appropriate corrections which eventually increased our accuracy rate. Experimental results and analysis have been demonstrated in Chapter 5.

4.2.3 Finding an error in complex, compound sentence structure

Since previous grammar checkers have a very small dataset, their system will not be able to detect different types of Bangla sentence structures such as simple, complex, compound and unorthodox which increases the error rate of the system.

As we have explained in Chapter 2, our corpus is a balanced corpus containing sentences from different sources and sentence types. Therefore, our system is capable of finding errors in different Bangla sentence structures.

4.2.4 POS Based n-gram model

According to our research, the most notable work in Bangla grammar checker was done by N. UzZaman and Mumit Khan where they used the language model probability on POS of the words [17]. He used an implementation of Brill's tagger with a training corpus of Bangla 5000 words and the accuracy rate of POS detection is only 50% [17]. If

the POS tagger gives us the accuracy rate of 50%, we cannot expect a grammar checker with high accuracy based on this POS tagger.

Although we found a very efficient Bangla POS tagger developed by Nahid Hossain and Mohammad Nurul Huda [20] with the accuracy rate of 91.45% based on a corpus of 23000 Bangla words, we did not use a POS tagger based language model probability instead we used a direct word-based language model probability. Because, first of all, if the POS tagger gives us a wrong POS of a word then language model probability will provide a wrong result. Secondly, finding probability based on POS of the word may lead us to a wrong decision. For example,

Table 17 : Comparison of POS based language model probability

SL	Sentence	Transliteration	POS
1	আমি ভাত খাই	ami bhat khai	PRN NN VB
2	আমি ভাত পড়ি	ami bhat pori	PRN NN VB

Table 17 demonstrates two different sentences where the first sentence is correct and the second one is incorrect. However, POS sequence of both sentences is the same. Since using a POS-based language model probability may lead to an incorrect decision, we have used the direct word-based language model probability.

4.2.5 No Suggestion for Grammatical Error

We observed all the grammar checker only detect errors without suggesting any solution for the grammatical mistakes. Suggesting a solution helps a user to correct the problem in an efficient way and reduces the time of correcting grammatical mistakes.

Our system not only detects grammatical mistakes but also suggest appropriate suggestion for those. We have used Cosine similarity to determine the closest match between corpus sentences and with the incorrect sentence, and then our system suggests two closest sentences as a tentative solution for the grammatically incorrect sentence.

There are other negligible limitations observed in the previous approaches. Most of the previous approaches neither solve these aforementioned issues nor mentioned or clarified these issues in their study. However, we have tried to solve each of these issues.

4.3 Algorithm and System Diagram of Grammar Checker

The system diagram of our proposed spell checker section has been given below.

Table 18 : Table of terminology (grammar checker section)

Term	Definition
DMV	Double Metaphone Values
DMA	Double Metaphone Algorithm
UED	Unicode-based Edit Distance
ED	Edit Distance
CS	Cosine Similarity

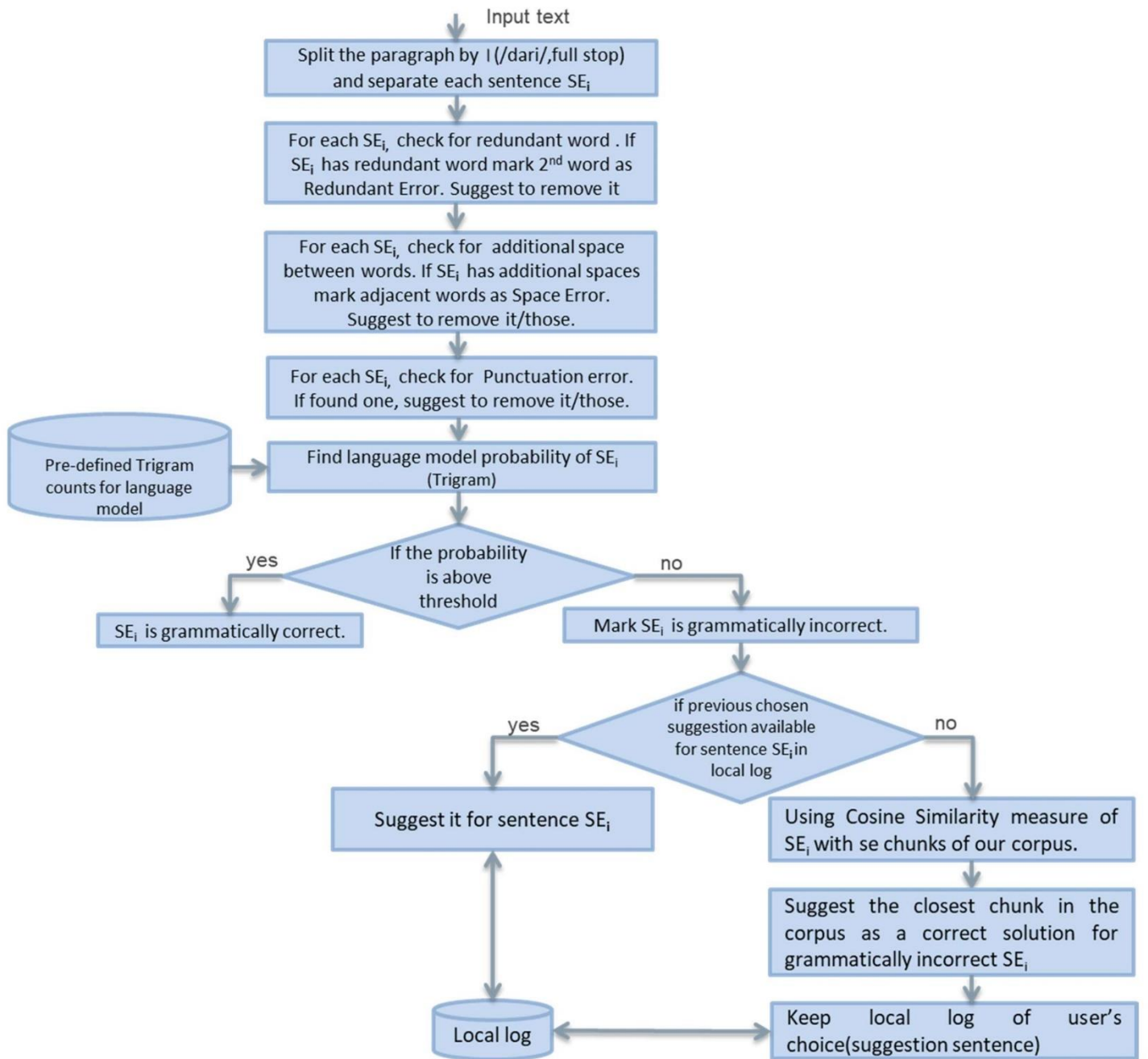


Diagram 2 : System diagram of grammar checker section

The algorithm of grammar checker is given below.

Algorithm 4 Grammar Checker

```
1. IN  $\leftarrow$  input text
2. SEi  $\leftarrow$  split IN into separate sentences using | (/dari/, full stop)
3. START BASIC CHECK
4.   IF SEi contains any redundant words consecutively
5.     Mark it as grammatical error and suggest to remove
6.   END IF
7.   IF SEi contains any additional whitespaces in between words
8.     Mark it as grammatical error and suggest to remove
9.   END IF
10.  IF basic punctuation error found(such as duplicate punctuation, missing
    punctuation)
11.    Mark it as grammatical error and suggest to remove or add punctuations
12.  END IF
13. END BASIC CHECK
14. TRi  $\leftarrow$  find language model probability (in our trigram) of SEi based on pre-
    calculated counts
15. IF TRi is above a threshold (in our case 0)
16.   Mark SEi as grammatically correct
17. ELSE
18.   Mark SEi as grammatically incorrect
19.   IF previous grammatical suggestion available for sentence SEi
20.     Suggest it for SEi
21.   ELSE
22.     Find Cosine Similarity(CS) between SEi and sentences in corpus
23.     Keep a track of closest sentences with SEi
24.     Display top two closest match as a potential grammatical correct sentence
    for SEi
25.     Keep local log of user's choice for future customized suggestion
26.   END IF
27. END IF
```

Why Trigram? We have used trigram in language model probability. At the beginning of our study, we decided to use bigram for simplicity. However, our study shows bigram failed miserably to detect grammatically incorrect sentences if a sentence is large or in a complex structure. Moreover, 4-gram or more than that increases processing time and requires more memory. In addition, it fails to detect errors in small sentences such sentences consists of two words only. Therefore, our study shows using the trigram model in language model probability increased accuracy in almost all cases. In addition, we have used pre-calculated trigram counts which reduce processing time vastly.

Why Cosine Similarity? For providing appropriate suggestions, we have used the cosine similarity algorithm to measure the similarity between the grammatically incorrect sentence and sentences in our corpus. At the very beginning of our study, we decided to use Levenshtein's edit distance(ED) instead of cosine similarity. Later, when we applied ED the result generated was not what we expected and our research shows ED was not giving the accurate result. Moreover, ED is not suitable for multi-byte characters like Bangla. We observed a very high processing time when we have used ED for Bangla sentences. Finally, after studying several similarity measure algorithms and different aspects of Bangla language, we have decided to use cosine similarity which provided both increased accuracy and decreased processing time.

Chapter 5

Experimental Results and Analysis

5.1 Spell Checker System

Here, we will show different experimental views of our spell checker based on different inputs. For simplicity, all examples will be on single sentences. Figure 2 demonstrates how our spell checker shows results if there is no spelling error.

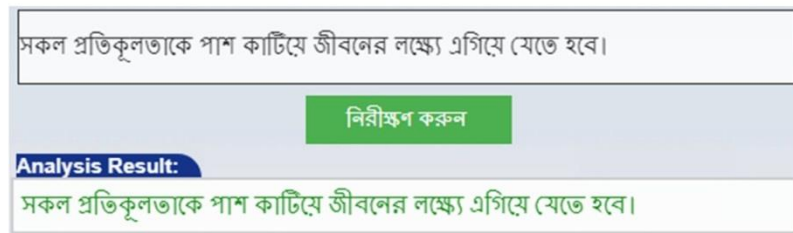


Figure 2 : Input has no spell error

In Figure 3, we have shown a sentence with two spelling errors. We wrote (first error from left side) ‘প্রতিকূলতাক’ instead of the correct word ‘প্রতিকূলতাকে’(TL: protikoolotake, T: adversity) and we wrote (last error from left side) ‘হভে’ instead of the correct word ‘হবে’(TL: hobe, T: have to).

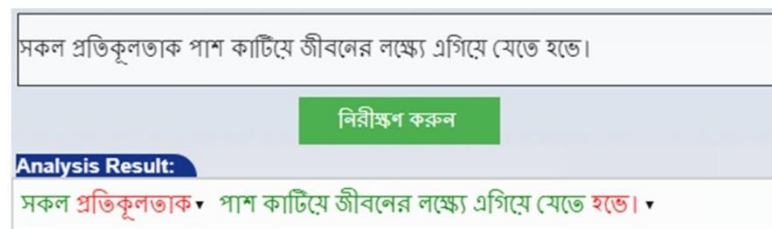


Figure 3 : Sentence with two spelling errors

Beside each error, we have a down arrow () to indicate there are suggestions for this wrong word. Figure 4 shows what happens when we click the arrow to see appropriate suggestions for the wrong word ‘হভে’(2nd one).

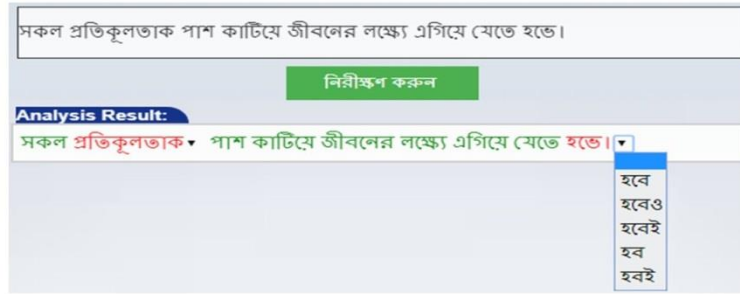


Figure 4 : Sentence with two spelling errors and suggestion

The suggestion shows five tentative correct words for the misspelt word ‘হতে’ where ‘হবে’ is the actual word. In this case, the actual suggestion word showed up at the top of the suggestion list. (Whenever a user clicks on the correct word ‘হবে’ from the suggestion list, our approach saves the information for future suggestion to increase improved user experience in a local log in the following format, ‘wrong_word : selected_correct_word : frequency’. Frequency means the number of times a user choose the selected_correct_word word as a correct word for the misspelt word wrong_word. For this example for the first time only our approach saves the following information in the local log, ‘হতে : হবে : 1’.). However, this is not the case always. Depending on several criteria(explained in section 3.6 in the algorithm) the actual correct word may not be found at the top of the suggestion list in every case. This situation has been explained using the first error of the same example sentence in Figure 5.

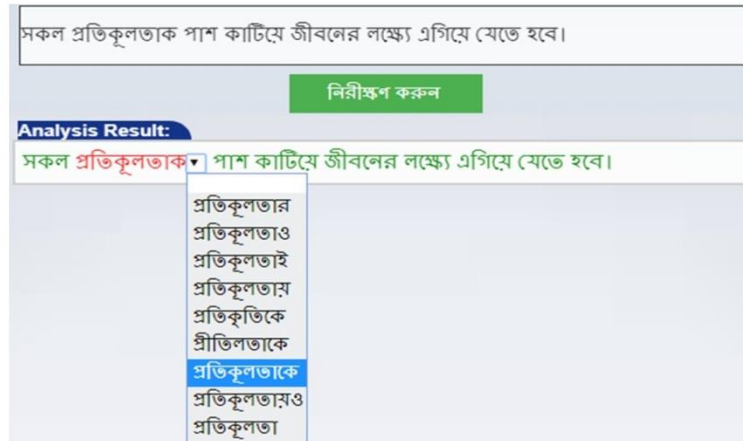


Figure 5 : List of suggestions where tentative correct word not at the top

In this case, there are several appropriate words suggested for the misspelt word ‘প্রতিকূলতাক’. However, the actual correct word would be ‘প্রতিকূলতাকে’ in this context

which is at the 7th position in the suggestion list. This may sometimes be inconvenient to users. We keep log of user's choice for future suggestions such as for this situation. When the user choose the 7th word 'প্রতিকূলতাকে' as correct word for the misspelt word 'প্রতিকূলতাক', we keep log of this word choice and in future when the user makes the same mistake again this time our approach suggest the correct word at the top of the list as shown in Figure 6.

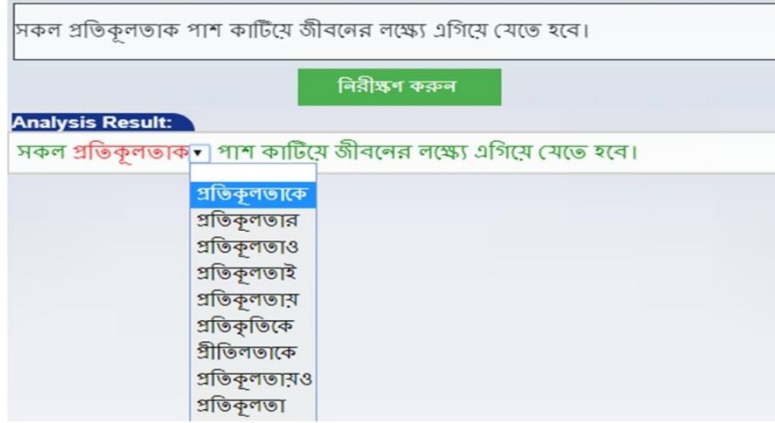


Figure 6 : Tentative correct word at the top based on previous log

However, if a user chooses different words at different times for the same misspelt word then our approach will show the most frequently chosen word at the top.

Handling foreign words and Bangla numbers are also very essential since we see the use of foreign words (i.e. English) and numbers(with or without text fragments) more often in Bangla text. In our approach, if we detect a foreign word using Unicode, we avoid processing the word and color the word with blue. Since our proposed method has been designed to find Bangla spell and grammar checker; it is obvious to avoid processing any foreign word. However, we make the color of the word blue instead of green to indicate that the word is unprocessed. Example of foreign word handling has been shown in Figure 7. Here, the word 'amazon' is a foreign word thus it kept unprocessed.

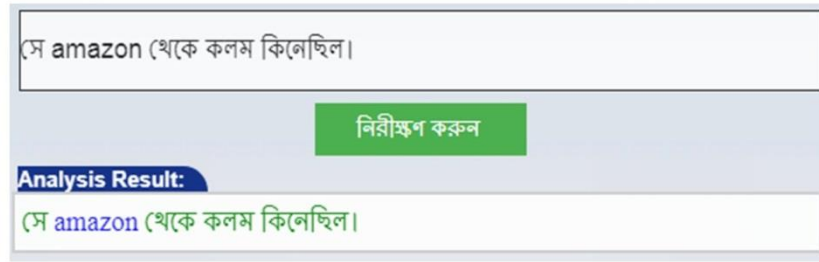


Figure 7 : Foreign words handling

On the other hand, if we detect a word where each and every character is Bangla number then we mark it as correct without further processing. Since variations in numbers are enormous; we mark it as correct if and only if the number has no character in it (figure 8 shows an example of it).

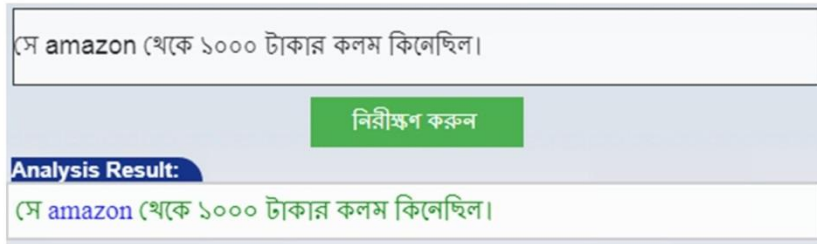


Figure 8 : Example of word '১০০০' which is purely numerical

However, if the word is not purely numerical containing one or more characters in it, there will be two possible way to detect errors. One, check if the character(s) placed in random places such as a number then a character and then another number such as ১টি৩ , ১২ক১, ক৫. If this is the case, then we mark it as incorrect word and find a suitable suggestion (as close as possible) for it.

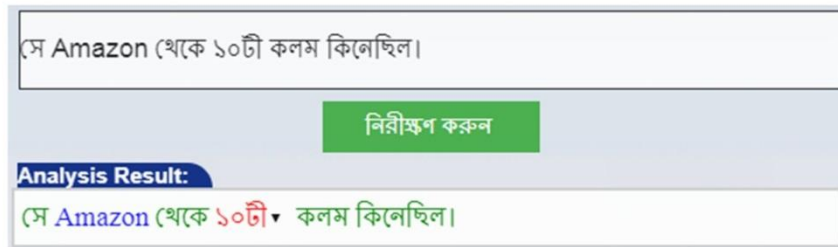


Figure 9 : Example of word '১০টি' which is alphanumerical and incorrect

Two, check if the character(s) has been placed at the end of the word with numbers at the beginning such as ১০টি (correct) and ১০ট (incorrect). In this case, we separate the characters from the numbers and check if the characters available in our data set. If yes, then the whole word is correct, otherwise incorrect and our approach shows a suitable suggestion for the incorrect word. Figure 9 shows an example of incorrect alphanumeric word detection and in figure 10 shows the tentative suggestions for it.

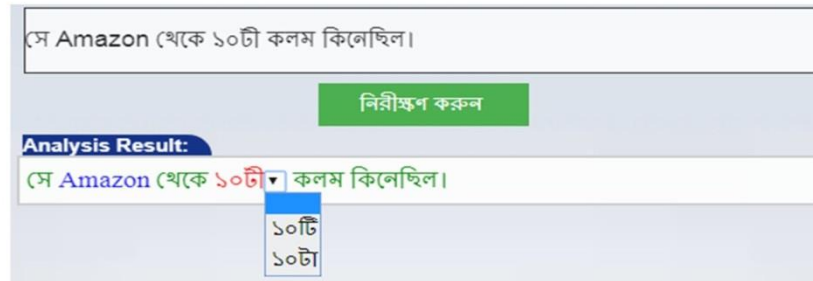


Figure 10 : Suggestion for the incorrect alphanumerical number.

5.2 Grammar Checker System

Here, we will show different experimental views of our grammar checker based on different inputs. For simplicity, all examples will be on single sentences. As we have mentioned earlier in our grammar checker system we first apply basic checks such as any existence of consecutive redundant words, duplication of punctuation and additional whitespaces. Figure 11 demonstrates how we handle consecutive redundant words.

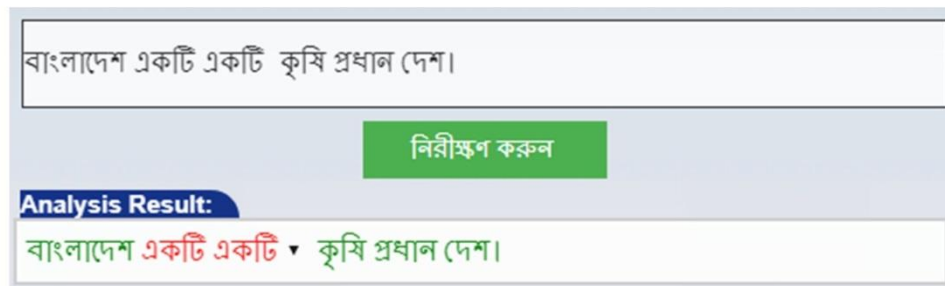


Figure 11 : Detecting consecutive redundant words

Here, we observed a consecutive redundant words ‘একটি’ which is grammatically incorrect and thus our system detect it as incorrect and suggestion of this problem in showed in figure 12.

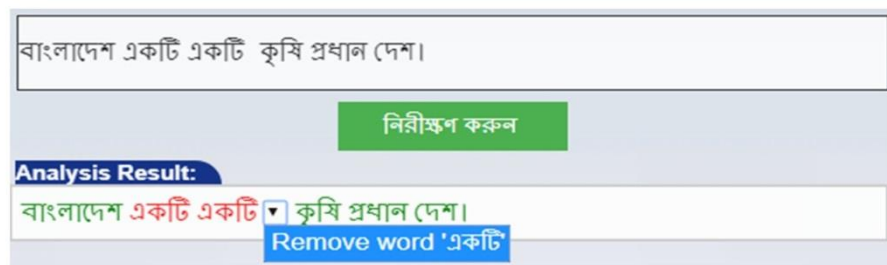


Figure 12 : Suggestion for the consecutive redundant words error

Now, if our system detects any additional whitespace(s) between words it marks that portion as incorrect and suggests removing it and when a user clicks on it will automatically remove the whitespace(s) just like it remove consecutive redundant words. Figure 13 shows the additional whitespace scenario.

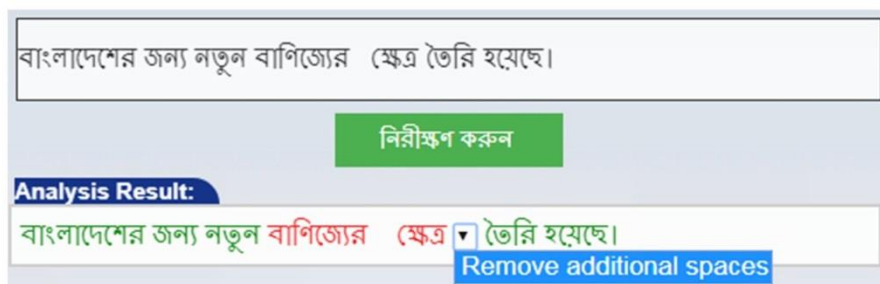


Figure 13 : Detecting additional whitespaces and Suggestion to remove

Putting punctuation in a wrong place is a common error in any article. Moreover, missing punctuation or adding extra punctuation is another common error in article writing. Figure 14 and Figure 15 shows how we detect and suggest extra punctuation or missing punctuation in an article respectively.

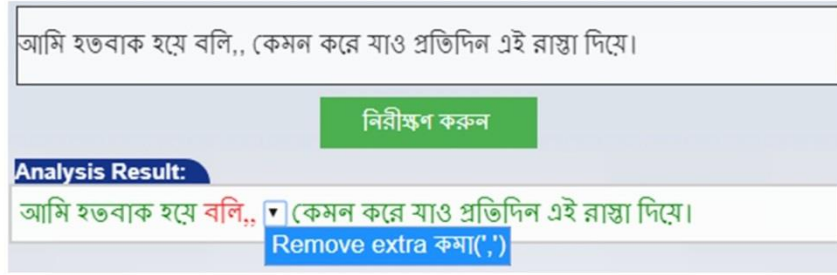


Figure 14 : Identifying punctuation error(extra)

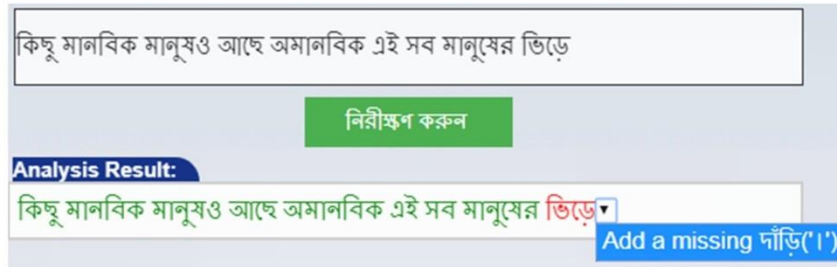


Figure 15 : Identifying punctuation error(missing)

Let's take a simple sentence 'আমি ভাত খাই।' (Transliteration: ami bhat khai) is a grammatically correct sentence. Now as we have mentioned the limitation of previous approaches (section 4.2), if someone makes a mistake in SV agreement such as 'আমি ভাত খায়' (Transliteration: ami bhat khay) ; the sentence is syntactically (POS Sequence) correct, therefore, any previous approaches would have told the sentence is valid. However, in our system, we have used direct word-based language model probability which can easily identify the difference. To suggest appropriate solution we have used the cosine similarity measure that provides two closest matches of any grammatically incorrect sentence as a tentative solution. Example is shown in Figure 16.

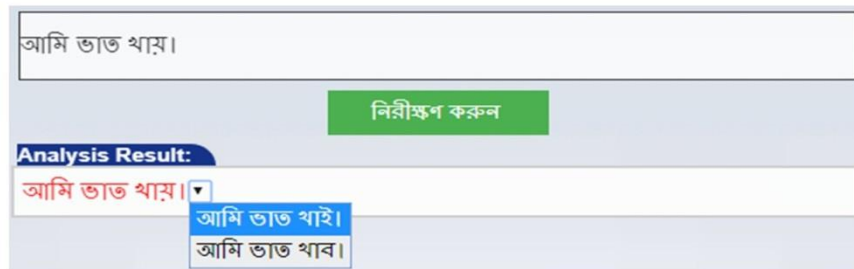


Figure 16: Identifying and suggesting a grammatically incorrect sentence(example-1)

Another example is given in Figure 17 where the sentence does not follow the SOV(subject object verb) sentence structure. Here, the word ‘থাকে’ is a verb and should be at the very end of a Bangla sentence.

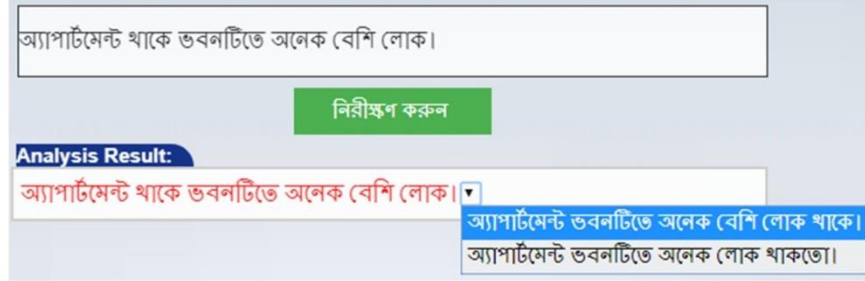


Figure 17 : Identifying and suggesting a grammatically incorrect sentence(example-2)

5.3 Performance Calculation

In order to calculate the performance of our system, we test our system with three different dataset which contains to different types of errors. First set of data contains sentences with spelling mistakes only. Second set of data contains only sentences that are grammatically incorrect all three aspects (syntactic, semantic and pragmatics) and the final set of data contains sentences with both spelling and grammatical mistakes.

Table 19 : Test dataset size

Data Set Type	Total sentences	Total words
Test-dataset-spell	8021	37235
Test-dataset-grammar	7545	34255
Test-dataset-both	8835	41236

Test-dataset-spell contains sentences with one or two misspelt word(s) in each sentence. Test-dataset-grammar contains at least one grammatical mistake in each sentence. Test-dataset-both contains both spell and grammatical errors in each sentence. Some sentences have only one type of error where others have multiple errors and/or combination of both. We have achieved a significant accuracy by our system. The test summary is given in Table 20.

Table 20 : Performance summary of our system

	Spell Checker	Grammar Checker	Spell and Grammar Checker
Dataset used	Test-dataset-spell	Test-dataset-grammar	Test-dataset-both
Total Sentences	8021	7545	8835
Recognized	7764	7114	8228
Not Recognized	257	431	607
Error Rate	3.2%	5.71%	6.87%
Accuracy Rate	96.8%	94.29%	93.13%

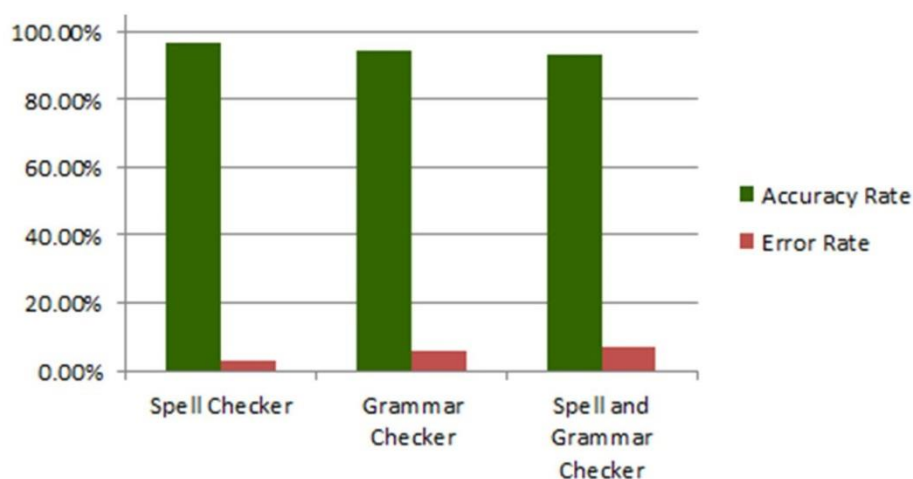


Figure 18 : Accuracy and Error rate presentation

i. Error Definition: We consider it fail if our system fails to detect all the errors in each sentence. We observed in some sentences it successfully detected and suggested an error however failed to detect all the errors in that sentence successfully. In such cases, we have considered it as a fail.

ii. Potential Reasons for Error: Although our system provides very high accuracy, our system still sometimes provides erroneous output. The output may wrong in both error detection or in giving appropriate suggestions. According to our analysis, there are several reasons behind this such as biasness in our corpus since we can't guarantee 100% words are spell and grammatical error free in our source of the corpus. Although we have automatically and manually filtered out numerous biased sentences, there is still some degree of biasness exists in the corpus. In addition to that, in spell checker, we have considered the words with a maximum edit distance of 1. However, in some cases, the correct word may exist with edit distance greater than 1. Moreover, for a very large and complex sentence, our algorithm may sometimes provide incorrect suggestions due to unavailability of similar sentences though this case is very rare.

Chapter 6

Conclusion, Limitations and Future Work

Our proposed system exhibits a Bangla spell and grammar checker. In order to develop a robust Bangla spell and grammar checker we have also developed the largest Bangla monolingual corpus and later we developed the largest Bangla monolingual lexicon by extracting unique lexemes from the corpus. We have demonstrated each and every step of building the corpus, lexicon and the spell and grammar checker. Finally, we have shown the system diagrams, algorithms and experimental results and analysis. We have explained the boundaries of previous approaches and provided solutions to those limitations. We have achieved the accuracy rate of 96.8% in the spell checker, 94.29% in the grammar checker section and 93.13% in both spell and grammar checker combined.

The limitation of the work is the biasness in our corpus. As we have mentioned in section 2.7, we have collected our texts from various reliable sources. However, we cannot guarantee that 100% of the texts are spell and grammatical error free.

The authors are working hard on it to increase the accuracy rate. For this reason, authors are studying the failed cases and are trying to fix our algorithm accordingly. Authors will contact the Bangla Academy (autonomous institution funded by the Bangladesh government to promote and preserve the Bangla language) authority for the formal accreditation of our corpus.

References

- [1] Wikipedia contributors, "Bengali language," Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=Bengali_language&oldid=881491578 (accessed February 5, 2019).
- [2] B.B. Chaudhuri, "Reversed word dictionary and phonetically similar word grouping based spellchecker to Bangla text", Proc. LESAL Workshop, Mumbai, 2001.
- [3] M.T. Haque and M. Kaykobad, "Use of Phonetic Similarity for Bangla Spell Checker", Proc. 5th International Conference on Computer and Information Technology, Dhaka, December, 2002, pp. 182-185.
- [4] N. UzZaman and M. Khan, "A Bangla Phonetic Encoding for Better Spelling Suggestion", Proc. 7th International Conference on Computer and Information Technology, Dhaka, Bangladesh, December, 2004.
- [5] N. UzZaman and M. Khan, "A Double Metaphone encoding for Bangla and its application in spelling checker," 2005 International Conference on Natural Language Processing and Knowledge Engineering, Wuhan, China, 2005, pp. 705-710.
- [6] M. Khan. "Creation and Analysis of a New Bangla Text Corpus BDNC01". 2017 International Journal for Research in Applied Science and Engineering Technology. V. 260-266. 10.22214/ijraset.2017.11038.
- [7] M. Mumin, A. Shueb, M. Selim and M. Iqbal, "SUMono: A Representative Modern Bengali Corpus" SUST Journal of Science and Technology, Vol. 21, No.1, 2014; P: 78-86

- [8] M. Khan and A. Sobhan. Article: Observations from Statistical Processing of BDNC01 Corpus. International Journal of Applied Information Systems 3(3):1-7, July 2012
- [9] Scrapy.org. (2019). Scrapy | A Fast and Powerful Scraping and Web Crawling Framework. [online] Available at: <https://scrapy.org/> [Accessed 5 Feb. 2019].
- [10] Python.org. (2019). Welcome to Python.org. [online] Available at: <https://www.python.org/> [Accessed 5 Feb. 2019].
- [11] Inspur Systems. (2019). NF5180M5 - Inspur Systems. [online] Available at: <https://www.inspursystems.com/product/nf5180m5> [Accessed 5 Feb. 2019].
- [12] Wikipedia contributors, "Lexicon," Wikipedia, The Free Encyclopedia, <https://en.wikipedia.org/w/index.php?title=Lexicon&oldid=862542401> (accessed February 5, 2019).
- [13] Wikipedia contributors, "Levenshtein distance," Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=Levenshtein_distance&oldid=880965610 (accessed February 5, 2019).
- [14] N. UzZaman and M. Khan, "A Comprehensive Bangla Spelling Checker" In the Proceeding of International Conference on Computer Processing on Bangla (ICCPB-2006), Dhaka, Bangladesh, 17 February, 2006.
- [15] F. DAMERAU, "A Technique for Computer Detection and Correction of Spelling Errors." In Communications of ACM, Vol. 7, No. 3, pp. 171- 177, March, 1964.
- [16] Z. Bhatti.(2012) "Spelling Error Trends and Patterns in Sindhi". Journal of Emerging Trends in Computing and Information Sciences. 3. 1435-1439.

- [17] M. Jahangir, N. Uzzaman and M. Khan. “N-Gram Based Statistical Grammar Checker For Bangla And English”. Center for Research On Bangla Language Processing. Bangladesh, 2006.
- [18] K. Hasan, A. Mahmud, A. Mondal & A. Saha. (2012). Recognizing Bangla Grammar Using Predictive Parser. International Journal of Computer Science and Information Technology. 3. 10.5121/ijcsit.2011.3605.
- [19] M. Haque and M. Kaykobad, “Coding System for Bangla Spell Checker”, Page 186 – 190, Proc. 5th International Conference on Computer and Information Technology, Dhaka, December, 2002
- [20] N. Hossain and M. Huda, “A Comprehensive Parts of Speech Tagger for Automatically Checked Valid Bengali Sentences”. International Conference on Computer and Information Technology(ICCIT) 2018. In Press.
- [21] S. Islam, M. Sarkar, T. Hussain, M. Hasan, D. Farid and S. Shatabda, “Bangla Sentence Correction Using Deep Neural Network Based Sequence to Sequence Learning”. International Conference on Computer and Information Technology(ICCIT) 2018. In Press.
- [22] S. Ahmed, T. Goncalves, and H. Sarwar. (2016). Improving Bangla OCR output through correction algorithms. 338-343. 10. 1109/ SKIMA. 2016. 7916243.
- [23] M. Biswas, R. Islam, G. Kumar Shom, M. Shopon, N. Mohammed, S Momen, and A. Abedin. (2017). BanglaLekha-Isolated: A multi-purpose comprehensive dataset of Handwritten Bangla Isolated characters. Data in Brief. 12. 103-107. 10.1016/j.dib.2017.03.035
- [24] H. Taherdoost. (2016). Sampling Methods in Research Methodology; How to Choose a Sampling Technique for Research. International Journal of Academic Research in Management. 5. 18-27. 10.2139/ssrn.3205035.