IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received September 19, 2021, accepted October 7, 2021, date of publication October 13, 2021, date of current version October 21, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3119627

Development of Bangla Spell and Grammar Checkers: Resource Creation and Evaluation

NAHID HOSSAIN, SALEKUL ISLAM, (Senior Member, IEEE), AND MOHAMMAD NURUL HUDA

Computer Science and Engineering Department, United International University, Dhaka 1212, Bangladesh

Corresponding author: Salekul Islam (salekul@cse.uiu.ac.bd)

ABSTRACT A spell and grammar checker is profoundly essential for diverse publications especially for Bangla language in particular as it is spoken by millions of native speakers around the world. Considering the lack of research efforts, we demonstrate the development of a comprehensive Bangla spell and grammar checker with necessary resources. At first, a full-fledged and generalised Bangla monolingual corpus comprising over 100 million words has been built by scraping reputed, diversified online sources and then an extensive Bangla lexicon consisting of over 1 million unique words has been extracted from that corpus. Based on these corpus and lexicon, we have developed a combined spell and grammar checker application that simultaneously detects distinct spelling and grammatical mistakes and provides appropriate suggestions for both as well. The spell checker uses the Double Metaphone algorithm and Edit distance based on the distributed lexicons and numerical suffix dataset to detect all types of Bangla spelling mistakes with an accuracy rate of 97.21% individually. The grammar checker detects errors based on language model probability i.e. combination of bigram and trigram, and generates suggestions based on the Cosine similarity measure with the accuracy rate of 94.29% individually. The datasets and codes used in this work are freely available at https://git.io/JzJ4w.

INDEX TERMS Bangla, corpus, grammar checker, lexicon, spell checker.

I. INTRODUCTION

Bangla is a language spoken by roughly 250 million native speakers around the world, mainly in Bangladesh and some regions of India [1]. Every day hundreds of books, magazines, and newspapers are being published in Bangla. In order to inscribe a high-quality article, an article must be free from spelling and grammatical errors. Spell checking is essential to ensure the quality of content and readability for readers as well. Moreover, an online article free from spelling mistakes gives better crawl-ability and indexing for search engines. Detecting erroneous words is not that difficult for a spell checker. However, suggesting the appropriate word(s) for an erroneous word is challenging for a spell checker, especially in non-Latin languages such as Bangla. Finding grammatical mistakes is also essential while checking for spelling mistakes. Suggesting an appropriate solution for a grammatical mistake is the most difficult challenge. Moreover, the combination of both spell and grammar checkers is essential for a fruitful Bangla article. Currently, available spell and grammar checkers in Bangla are not prolific due to the very trivial dataset and primeval way of processing data.

In order to develop a functional spell and grammar checker, we need to have a comprehensive monolingual corpus and lexicon. Although extensive researches have been carried out focusing on spell and grammar checkers in English, very few researches have been found in Bangla. By studying the available Bangla corpus, lexicon, spell, and grammar checker, we have identified several limitations in the current approaches, including scarcity of balanced and extensive corpus, substantial lexicon, and efficient spell and grammar checker. A text corpus is a large and structured set of texts [2]. The corpus-based method of information processing gives language researchers the advantage of not relying on native speakers' or on their perceptions. There are incessant research efforts to practice corpus-based methods as the leading part of any language processing system. Since our proposed grammar checker is data-driven, we need to build a large, balanced, and well-defined Bangla monolingual corpus to make our grammar checker accurate. Later, we have extracted the corpus to construct the Bangla monolingual lexicon to enhance the spell checker's performance.

The associate editor coordinating the review of this manuscript and approving it for publication was Sun-Yuan Hsieh.

Everyday, hundreds of Bangla articles are being published through online and printed newspapers. Besides, thousands of Bangla books are being published every year. In social media such as Facebook and Twitter, millions of users are communicating with each other in Bangla every day. Moreover, there are many government and non-government places where documents are being written in Bangla. All these above-mentioned places will be benefited if there exists any functional spell and grammar checker by improving the quality of Bangla articles significantly. Moreover, a reliable, balanced, and large corpus is the root of many statistical-based linguistics research. This motivates us to develop a comprehensive spell and grammar checker along with a corpus and a lexicon for the Bangla language.

The notable research challenge we face while working on the Bangla language is the linguistic complexities of the language. Bangla has 11 vowels, 30 consonants, 10 diacritic forms of vowels, and a distinctive horizontal line above each alphabet. Moreover, it creates multiple other conjoint characters by joining two to three base characters. It has pairs of alphabets that look and pronounce nearly the same. Furthermore, word order in Bangla sentences affects the comprehension of a sentence in many cases. Unlike other well-studied languages, Bangla language has many distinctive linguistic features in word formations, sentence structures and unique word usages that make the language challenging to work with. We have tackled the challenge by in-depth analysis of the language, finding out appropriate solutions, and using different statistical approaches. Note that enormous supporting data helps to understand the complex linguistic features without having in-depth knowledge of the language in many cases.

To address the aforementioned research gaps, our objectives are summarised as follows. First, due to the lack of a sufficiently large corpus and a lexicon in Bangla for statistical linguistic analysis, we plan to develop a corpus and a lexicon. Second, to develop a reliable and accurate Bangla spell and grammar checker based on the corpus and lexicon we will develop. Finally, to publish our datasets (i.e., the corpus and lexicon) online to make available for the future researchers.

The significant contributions of our work are summarised in the following:

- Developing the largest (to the best of our knowledge) Bangla monolingual balanced corpus and lexicon consist of over 100 million words and one million words, respectively.
- Developing a comprehensive Bangla spell checker approach considering all types of Bangla spelling errors. The spell checker also provides appropriate suggestions for a misspelt word based on different criteria with the help of encoding values from Double Metaphone Algorithm [3], and Edit Distance [4] of a lexeme.
- Developing a full-fledged and generalised Bangla grammar checker approach considering various Bangla grammatical mistakes. The grammar checker provides appropriate suggestions to correct the mistake based on language model probability i.e. combination of bigram and trigram, and text similarity approach i.e. Cosine similarity measure.
- Developing a combined spell and grammar checker application that detects errors and shows suggestions on both spelling and grammatical mistakes in Bangla sentences unlike all previous approaches. Moreover, the achieved accuracy is higher than any previous approaches for both spell and grammar checkers.

The rest of the paper is organised as follows: section II describes the related works and identifies the limitations of previous approaches. The architecture of the proposed system has been briefly explained and described with a process diagram in section III. Section IV presents the step-by-step development process of the corpus and demonstrates the procedure of the extraction and development of the lexicon. Although we have developed a combined solution for spell and grammar checkers, the details of these two are presented in different sections for simplicity and better understanding. Section V describes the proposed spell checker, including the algorithm. The next section presents the proposed grammar checker. Section VII demonstrates the experimental results and the analysis of the results as well. Finally, section VIII concludes the paper by mentioning future work.

II. BACKGROUND

This section mentions the related works of three interconnected but distinct segments of our proposed system: corpus and lexicon, spell checker, and grammar checker.

A. CORPUS AND LEXICON

A corpus is a collection of written texts, especially the entire works of a particular author or writing body on a particular subject. On the other hand, a lexicon is a vocabulary, a collection of words, or a complete set of meaningful units in a language. Therefore, a lexicon is generally built from a generously large corpus.

Central Institute of Indian Languages (CIIL) [5] first introduced a Bengali corpus along with a corpus of other nine Indian languages in 2001. It contains 3 million Bangla words, which is not sufficient for today's large-scale applications. In 2006, a corpus named 'Prothom-Alo' [6] had been developed by collecting data from a leading Bangladeshi daily newspaper, named 'Prothom-Alo'. The corpus contains more than 18 million words. Although the size of the corpus is moderate, the corpus is not suitable for real-life applications since it collected texts from a single source that may provide imbalanced or biased data. In 2014, Mumin *et al.* [2] proposed a Bangla monolingual corpus, named 'SUMono', which contains 27 million words collected from various sources. At present, this is the largest Bangla monolingual corpus. Khan [7] proposed a Bangla monolingual corpus, named 'BDNC01' in 2017, which contains 12 million words. The author proposed the same monolingual corpus earlier in 2012 with 11 million words [8]. In 2020, Ahmed *et al.* [9] illustrated the building technique of a Bangla speech corpus from publicly available audio. In addition to that, they proposed a text corpus for their speech corpus with 10 million sentences. However, the authors neither mentioned how and from where they collected these data nor how they filtered and processed the text corpus. On the other hand, there are some other small-sized corpora available for different goals in Bangla. In 2012, Goldhahn *et al.* [10] proposed a collection of monolingual corpora for 200 languages, including Bangla, with some Bengali articles crawled from the internet. Biswas *et al.* [11] proposed a handwritten character dataset in 2017. Later in 2018, Alam and Islam [12] developed a dataset containing different Bangla articles for their article classification approach.

Table 1 shows the comparison among the available Bangla monolingual corpora. It shows that except our proposed corpus NHMono01, which consists of more than 100 million words, the sizes of all other corpora are not sufficient for comprehensive statistical linguistics research.

TABLE 1. Comparison of different Bangla monolingual text corpora.

Corpus Name	Total Words
NHMono01(Proposed)	100,142,522
SuMono	27,118,025
Prothom-Alo	18,100,378
BDNC01	11,362,524
CIIL	3,044,573

On the other hand, to the best of our knowledge, no lexicon is available for the Bangla language at this moment, and hence, the lexicon developed in this study is the first of its kind.

B. SPELL CHECKER

A spell checker is a computer program that checks the spelling of words in a text, typically by comparing with a stored list of words.

Haque and Kaykobad [13] proposed the Bangla phonetic encoding for spell checker based on the Soundex algorithm in 2002. They numbered each group of phonetically similar consonant characters with 1 to 9 and vowels with 0. Note that this Bangla Soundex table exhibits several imperfections, such as poor encoding of similar characters. Later in 2004, Uzzaman and Khan [14] projected a phonetic encoding based on the Soundex algorithm, which abrogated several limitations of Haque and Kaykobad's encoding. Uzzaman and Khan [3] proposed another spell checker based on the Double Metaphone Algorithm (DMA) in 2005, where the authors proposed a Double Metaphone encoding technique that improved the spell checker significantly. Later in 2006, the authors proposed a spell checker based on their previous approach, claiming a better suggestion for the misspelt words [15]. In 2016, Ahmed *et al.* [16] used the Edit Distance and n-gram models to post-process their Optical Character Recognition (OCR) output by omitting the misspelt words. Mandal and Hossain [17] proposed a clustering-based Bangla spell checker in 2017 that can handle only typographic and phonetic errors. They have used 2,450 misspelt words as their dataset. In 2018, Sooraj *et al.* [18] proposed a spell checker for the Malayalam language based on a neural network. Later in 2019, Mittra *et al.* [19] developed a Bangla spell checking technique to facilitate error correction in the text entry environment. They used 50,000 n-gram sentences to detect and correct the errors. Hasan *et al.* [20] proposed a solution for Bangla Speech to Text(STT) conversion approach in 2020. They also integrated a spell corrector to improve accuracy for the proposed STT system. It can handle particularly typographic errors. The authors did not mention any specific description for the spell checker. Recently in 2021, Ahamed *et al.* [21] proposed a spell corrector for the Bangla language using Norvig's algorithm and Jaro-Winkler distance which can handle specifically typographic errors. In the same year, Noshin Jahan *et al.* [22] proposed a hybrid model for Bangla word error detection correct that is based on bidirectional LSTM and bigram. They developed four small-sized corpora to train and test their approach and provides an accuracy rate of 82.86%.

C. GRAMMAR CHECKER

A grammar checker is a program or part of a program that finds out any possible grammatical mistakes from a given text. The text might be a single sentence or a collection of several sentences.

In 2006, Alam *et al.* [23] 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 Part-Of-Speech (POS) tagger. They used a dataset that consists of only 5,000 words and gained an accuracy rate of 53.7%. In 2012, Hasan *et al.* [24] had projected a Context-Free Grammar (CFG) for Bangla language and hence based on that grammar, they developed a parse tree with 225 sentences. According to the authors, this parser can be applied to the Bangla grammar checker to parse different Bangla sentences. In the same year, Islam *et al.* [25] proposed an approach that recognises correct Bangla sentences and rejects incorrect ones based on the Head-Driven Phrase Structure Grammar (HPSG). Rahman *et al.* [26] proposed an investigative design-based statistical approach based on n-gram for determining Bangla sentence validity. In the same year, Rabbi *et al.* [27] described a parsing technique for Bangla grammar recognition. It takes the CFG of the Bangla language as input and constructs a parser table from the grammar. In 2017, Rahman *et al.* [28] designed a Bangla parser using the Non-Deterministic Push-down Automata (NPDA). The NPDA parser takes a CFG of Bangla Language for preprocessing, and the parser can parse all forms of Bangla sentences. Islam *et al.* [29] proposed a Machine Learning-based Bangla sentence correction approach in 2018 that uses the sequence to sequence learning, contains a dataset of 250K sentences, and shows an accuracy rate of 79%. In 2020, Shetu *et al.* [30] proposed a grammar checker that handles a specific problem in Bangla article called Guruchondali Dosh, which means mixing of two Bangla writing styles, i.e., Sadhu and Chalito, and achieved an accuracy rate of 74.5%. Recently in 2021, Saha Prapty *et al.* [31] developed a rule-based parsing for Bangla grammar pattern detection. They developed a domain-specific CFG based on the rules of Bangla grammar and applied it to that domain.

D. LIMITATIONS OF PREVIOUS APPROACHES

In the following we have summarised the limitations of the previous approaches and also mentioned how we have handled those in our work:

- **Small corpus size and manual corpus building:** All the previous Bangla corpus creation approaches [5]–[12] suffer from small corpus size (mentioned in Table 1) and thus, are not suitable for sophisticated statistics-based linguistics research. Moreover, these approaches take long time in corpus creation due to manual techniques including copy-and-paste or typing printed documents. Typing articles may also introduce typographic errors in the corpus. We solve these limitations by introducing automatic crawling and scrapping in a controlled manner that leads us to collect large, error-free data within a short span of time.
- **Limited error detection and suggestions by spell checker:** The previous spell checkers including the recent ones ([20] and [21]) can detect only typographic errors. On the contrary, [22] can detect only real-word errors. Note that our approach can detect all types of known errors in Bangla spelling. For better suggestions of the frequent spelling mistakes, phonetic encoding algorithms such as Double Metaphone are very effective. However, only [3], [13] and [14] use phonetic encoding. Our approach uses the Double Metaphone algorithm to offer better suggestions for errors including errors that occur due to phonetically similar Bangla alphabets.
- **Small lexicon size:** In dictionary loop-up approaches lexicon size is another factor behind the performance of the system. However, the sizes of the lexicon in the previous approaches are comparably not suitable for accurate results and suggestions. Our spell checker uses a large lexicon extracted from our corpus that helps to detect diversified spelling mistakes and to present accurate suggestions for erroneous words. Another flaw in the previous systems except [20] is that they tested the systems using test datasets consisting of single words instead of complete sentences that might not reflect real-life scenarios.
- **Grammar checker without suggestions:** All the previous Bangla grammar checkers except [26] are based on small corpora which is the leading reason behind the low accuracy of most of the previous grammar checkers. Similarly, these approaches detect grammatical mistakes without suggesting any solution for the mistakes. The proposed grammar checker detects mistakes and suggests appropriate solutions for the mistakes. Examples are shown in Table 22 which are based on the proposed corpus.
- **Combined spell and grammar checker solution:** The previous approaches focus on the partial problem and therefore, proposed solutions either for Bangla spell checker or for Bangla grammar checker but not together. However, a high-quality article should be free from both spelling and grammatical errors. On the other hand, our proposed system detects both types of errors and provides suggestions for both as well.

III. PROPOSED WORK

This section presents the abstract view of the proposed work. Figure 1 shows the architectural diagram that demonstrates the abstract view of the proposed work and also describes how each components are connected to each other.

In the first phase, we develop a balanced corpus with sufficiently large size for statistical use. We are very cautious in each step from text domain selection to quality assurance. Next, we develop a lexicon by extracting unique words from the corpus. The lexicon contains both root words as well as words with bound morphemes.

We develop the spell and grammar checker that can work simultaneously. The spell checker removes all the punctuation's from a given text, splits the text into separate lexemes and processes each lexeme with the help of the numerical suffix dataset (for Bangla numbers with suffixes), and the lexicon (for all other lexemes). To provide an appropriate suggestion for an error, it uses edit distance between Double Metaphone Values (DMV) (generated by the Bangla character phonetic encoding through double metaphone algorithm) of the erroneous lexeme and the pre-calculated DMV values of each lexeme in the lexicon. This gives us a temporary list of suggestion lexemes, from which the top five lexemes based on several criteria are selected as suggestions.

On the other hand, the grammar checker splits the given text into sentences. For each sentence, the grammar checker first applies a basic check that includes redundant or duplicate lexeme check, additional whitespaces check, and punctuation error check (extra, missing or misplaced). The grammar checker then generates language model probabilities based on bigram or trigram counts depending on the lexeme count in the sentence. It decides the correctness of a particular sentence based on a threshold of the language model probability. Then it applies the Cosine Similarity measure between the sentence and the corpus, stores the scores temporarily, and displays the two closest results as tentative grammatically correct sentences.

In the following sections, we describe each component of this proposed system in details.

IV. DEVELOPMENT OF DATASETS: CORPUS AND LEXICON

Our datasets are comprised of the corpus and the lexicon we have developed. This section explains different steps such

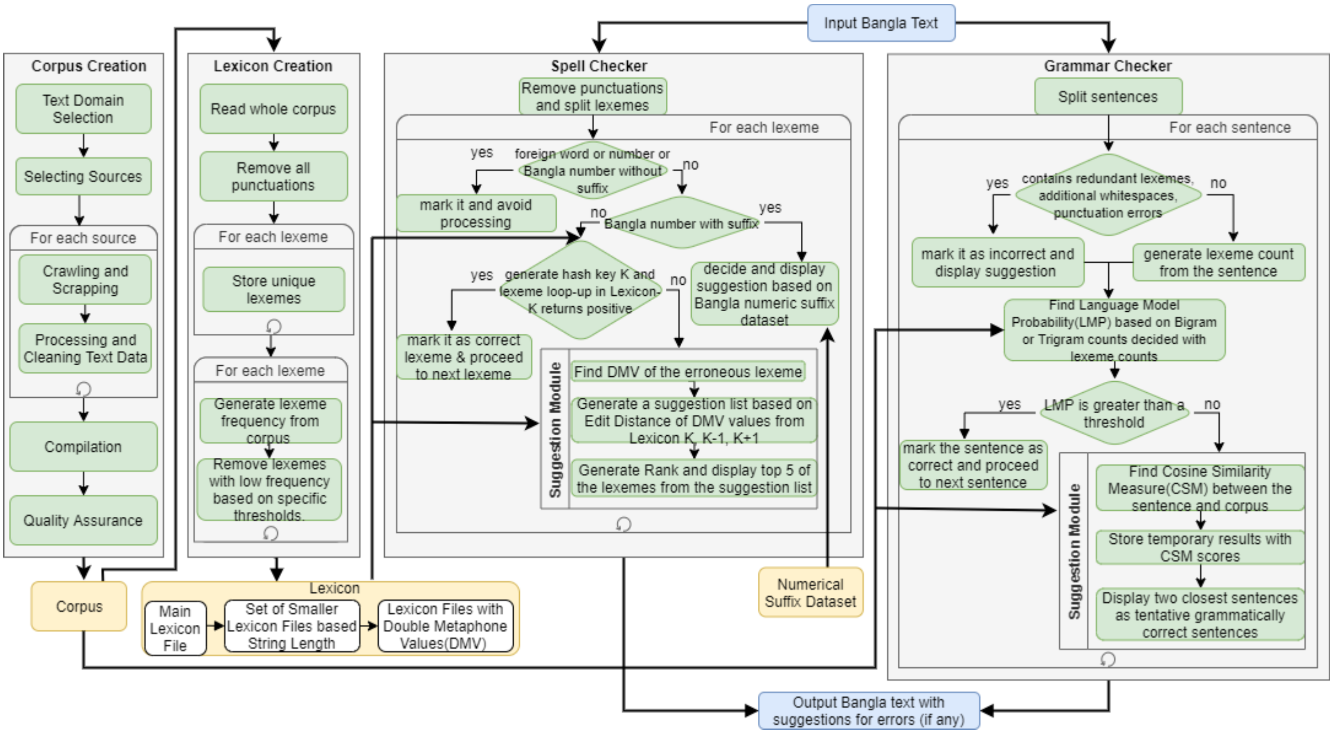
Input Bangla Text
Corpus Creation
Text Domain Selection
Selecting Sources
For each source
Crawling and Scrapping
Processing and Cleaning Text Data
Compilation
Quality Assurance
Lexicon Creation
Read whole corpus
Remove all punctuations
For each lexeme
Store unique lexemes
For each lexeme
Generate lexeme frequency from corpus
Remove lexemes with low frequency based on specific thresholds
Spell Checker
Remove punctuations and split lexemes
For each lexeme
yes
foreign word or number or Bangla number without suffix
no
mark it and avoid processing
no
Bangla number with suffix
yes
yes
generate hash key K and lexeme loop-up in Lexicon-K returns positive
no
decide and display suggestion based on Bangla numeric suffix dataset
mark it as correct lexeme & proceed to next lexeme
Suggestion Module
Find DMV of the erroneous lexeme
Generate a suggestion list based on Edit Distance of DMV values from Lexicon K, K-1, K+1
Generate Rank and display top 5 of the lexemes from the suggestion list
Grammar Checker
Split sentences
For each sentence
yes
contains redundant lexemes, additional whitespaces, punctuation errors
no
mark it as incorrect and display suggestion
generate lexeme count from the sentence
Find Language Model Probability(LMP) based on Bigram or Trigram counts decided with lexeme counts
yes
LMP is greater than a threshold
no
mark the sentence as correct and proceed to next sentence
Suggestion Module
Find Cosine Similarity Measure(CSM) between the sentence and corpus
Store temporary results with CSM scores
Display two closest sentences as tentative grammatically correct sentences
Corpus
Lexicon
Main Lexicon File
Set of Smaller Lexicon Files based String Length
Lexicon Files with Double Metaphone Values(DMV)
Numerical Suffix Dataset
Output Bangla text with suggestions for errors (if any)

FIGURE 1. Architecture of the proposed work.

TABLE 2. Domains of corpus.

SL	Domain
1	Newspapers
2	Science Magazines
3	Personal Blogs
4	Textbooks
5	Juvenile Literature
6	History
7	Poetry

as planning, analysis, design, and creation of corpus and lexicon development. Python 3.6 is the core language of these experiments with the scikit-learn library [32].

A. TEXT DOMAIN SELECTION

Although a corpus-based method is a good technique for language processing, it has some limitations as well. When a system works based on a particular corpus, it behaves according to that corpus. Therefore, if the corpus is not balanced, the performance degrades. When a corpus contains data from all possible domains, it shows balanced behaviour. Therefore, while collecting input data, we encompass the major domains that generate Bangla texts. The source domains that we have included are mentioned in Table 2.

B. SELECTING SOURCES

Each domain consists of numerous sources to collect data from the Internet. However, Table 3 shows the list of sources chosen to collect data based on our careful considerations and the reputation of different publicly available sources to increase the representativeness of the corpus.

TABLE 3. Domain and sources of corpus.

Domain	Sources	URL
Newspapers	Prothom Alo Bangla News 24 Kaler Kantho BD News 24	www.prothomalo.com www.banglanews24.com www.kalerkantho.com bangla.bdnews24.com
Science Magazines	Tech Tunes C News Voice	www.techtunes.co www.cnewsvoice.com
Personal Blogs	Somewhere in blog Sachalayatan	www.somewhereinblog.net www.sachalayatan.com
Textbooks	National Curriculum and Textbook Board	www.nctb.gov.bd
Juvenile Literature	Pandulipi	www.pandulipi.net
History	Banglapedia	bn.banglapedia.org
Poetry	Bangla Kobita	www.bangla-kobita.com

C. CRAWLING AND SCRAPING

Our approach for collecting text data is to crawl and scrape sources online. Crawling and scraping is the most advanced and fastest way to collect data. Using a spider bot to crawl and scrape is the automated way to accumulate information or text from a website. Python's Scrapy framework[1] is used to build the spider bots. Scrapy is an open-source framework for extracting data from any website. Since each source website has a different internal structure than others, we have to design and implement customised spider bot for individual sources by modifying the spider program accordingly. We have used a high-performance machine (hardware specification is given in subsection VII-B) to scrape and store data non-stop. The minutiae code snippet below illustrates the core segment of the spider that has been used to crawl and scrape data from the first source of the first domain of Table 3.

```
import scrapy
  class SpiderProthomAloSpider(scrapy.Spider
      ):
    name = ’SpiderProthomAlo’
    allowed_domains = [’www.prothomalo.com’]
    start_urls = [’http://www.prothomalo.com
        /archive/2020-07-24’]
    def parse(self, response):
        item_link = response.css(’a.
            link_overlay::attr(href)’).
            extract()
        file_object = open(”scrapped\_data\
            _prothomalo.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)
```

In this particular spider, in the `allowed_domains` variable, we put the URL of the target website (in this case, 'www.prothomalo.com'). Then, we mention the URL to start scraping in `start_urls` variable. The spider automatically crawls all the articles available from the date mentioned in the `start_urls` to the last available article/page. Inside a loop, we extract all the text contents only contained in the main body paragraph(s) from each page and append the texts into a Unicode text file (in this case '`scrapped_data_prothomalo`'). Each spider uses the HTML stripping technique to avoid all other unnecessary texts, such as the texts in titles, menus, advertisements, and any text in the header and footer. We crawl, scrape, and store Bangla texts from all other source websites in the same way with slight modification in the code of the spiders.

D. PROCESSING AND CLEANING OF TEXT DATA

We have collected roughly 4.0 gigabytes of plain Bangla texts in separate Unicode text files for each source. Although the spiders have been specifically designed to collect only plain Bangla body text from articles/pages, sometimes they have collected very few HTML tags, a lot of blank spaces and newlines, and most importantly, some English sentences. Therefore, in order to have pure Bangla plain text, each individual text file has been gone through the following steps:

- Apply further HTML stripping to remove any left-out HTML tags in the texts.
- Remove all blank spaces and newlines from each file.
- Remove additional whitespaces from each sentence.
- Remove any English words, numbers, or sentences from each file. English words have been identified based on Unicode code point.[2]
- Remove boilerplate texts.
- Remove duplicate sentences. In some sources, the exact same sentences can be repeated multiple times, especially in newspapers. To avoid negative effects in statistical analysis, they are discarded.

Nearly 2.0 gigabytes of pure Bangla plain text are left after cleaning and removing almost 2.0 gigabytes of useless data. Table 4 shows some samples of cleaning data.

E. COMPILATION AND TIME PERIOD

We have compiled all the individual text files into a single text file of nearly 2.0 gigabytes. During the compilation process, we have further removed duplicate sentences arrived from different sources. This happened especially in newspapers since reports of a specific incident appear almost the same in different newspapers. We have then stored both individual text files and the single compiled file in `CSV` and `TXT` formats. As a result of the continuous effort of seven months, we have developed the corpus and named it 'NHMono01'. The final version of the corpus consists of over 100 million words, 100,142,522 words, to be precise.

Total corpus creation time can be divided into two main phases. The first phase consists of selecting domains and sources, development of spiders, crawling and scraping. The second phase consists of post-processing, such as cleaning, removing inappropriate data, sorting data, compiling, generating statistics, and quality assurance. It took roughly seven months to complete the whole task. The corpus still needs continuous attention for any minor revisions in the corpus.

F. QUALITY ASSURANCE

Quality assurance is a major challenge as hundreds of corpora of up to millions of sentences can hardly be evaluated by hand [10], [33]. Thus, it must be a goal to do as little manual evaluation as possible. General features of natural language texts, such as empirical language laws (e.g., Zipf's law [34]) and behavior of function words to measure homogeneity of

[1] https://scrapy.org/

[2] https://www.unicode.org/charts/

TABLE 4. Samples of cleaning data.

Sample Raw Text Input	Output Text After Cleaning	Removal Procedure
চা পানের জন্য লাগে না কোনো বাহানা কিংবা সময়। <b> সকাল, দুপুর, বিকেল কিংবা রাত, চায়ের আড্ডা চলতেই থাকে। </b> চায়ে যোগ করতে পারেন বাড়তি কিছু উপাদান। This recipe is also available in English.	চা পানের জন্য লাগে না কোনো বাহানা কিংবা সময়। সকাল, দুপুর, বিকেল কিংবা রাত, চায়ের আড্ডা চলতেই থাকে। চায়ে যোগ করতে পারেন বাড়তি কিছু উপাদান।	*HTML tag removal, Additional whitespaces removal, English sentence removal*
জুতা সঠিক হলে দৌড়ানো যায় অনেকক্ষন। আরাম ও স্বস্তিও দেবে। পরবর্তীতে পা ব্যথা করবে না।	জুতা সঠিক হলে দৌড়ানো যায় অনেকক্ষন। আরাম ও স্বস্তিও দেবে। পরবর্তীতে পা ব্যথা করবে না।	*Additional newline removal, Additional whitespaces removal*
গুণগত মান ঠিক রেখে নির্ধারিত সময়ে বাসা হস্তান্তর করার ক্ষেত্রে তারা অনেক এগিয়ে। বরাবরই নির্দিষ্ট তারিখে হস্তান্তরের অঙ্গীকার বিটিআই পূরণ করে চলেছে। গুণগত মান ঠিক রেখে নির্ধারিত সময়ে বাসা হস্তান্তর করার ক্ষেত্রে তারা অনেক এগিয়ে। স্বত্ব © প্রথম আলো সম্পাদক ও প্রকাশক: মতিউর	গুণগত মান ঠিক রেখে নির্ধারিত সময়ে বাসা হস্তান্তর করার ক্ষেত্রে তারা অনেক এগিয়ে। বরাবরই নির্দিষ্ট তারিখে হস্তান্তরের অঙ্গীকার বিটিআই পূরণ করে চলেছে।	*Duplicate sentences removal, Boilerplate texts removal*

the corpus are the possible indicators of the quality of a corpus.

1) ZIPF'S LAW

For any corpus, the first and simplest query is a list of words, which can be organised by the frequency order. The frequencies follow Zipf's Law, which states that for a corpus of a natural language, the frequency of any word is inversely proportional to its rank. It is not known why Zipf's law holds for most of the languages. The Zipf's curve appears approximately linear on the log-log plot for most of the standard corpora. Zipf's law can be represented through the following approximation of the rank-frequency relationship:

$$rf = c \tag{1}$$

where r is the rank of a word-type, f is the frequency of occurrence of the word-type, and c is a constant. When stated algebraically, Zipf's law is usually given in the form of equation 1, but the law is probably most familiar in a graphical representation of the mathematically equivalent form:

$$log(r) + log(f) = log(c) \tag{2}$$

Equation 2 can also be written as follows:

$$\alpha(log(r)) + log(f) = log(c) \tag{3}$$

where on a log-log scale the plotted line will be a straight line with a negative slope α close to -1.

Figure 2 shows the Zipf's curve for our corpus NHMono01, and it is almost linear. The analysis of the graph shows that the term distribution in the whole dataset fits Zipf's law comfortably.

2) DISTRIBUTION OF FUNCTION WORDS

Function words have ambiguous meaning and express grammatical relationships among other words within a sentence. Function words are not informative unlike content words in a document. However, function words can be used to assess the quality of a corpus. In a balanced corpus, the function words will tend to distribute more homogeneously than content words [2], [35].

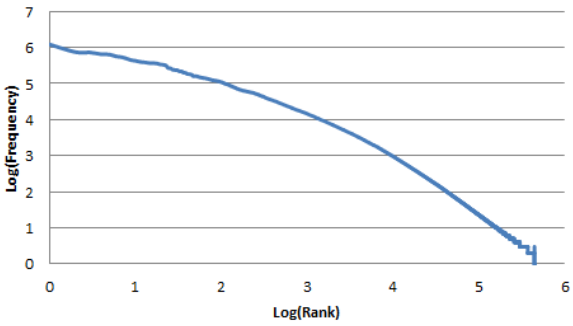

FIGURE 2. Plotting of word frequencies against its ranks to observe the Zipf's Law.

We have distributed the corpus into three groups to investigate the distribution of function words in the corpus. Each group contains data from two or more domains. The groups are as follows:

- Group-1: Newspapers, Science Magazines
- Group-2: Personal Blogs, Textbooks
- Group-3: Juvenile Literature, History, Poetry

Table 5 shows the 10 most frequent function words from each group and their percentage of occurrences. The table demonstrates that the most frequent function words are nearly the same words in all 3 groups with a slight difference in the position or rank of the words. The homogeneous distribution of these frequent function words confirms that the proposed corpus is well-balanced.

On the other hand, as mentioned in subsection IV-B, we have selected very well-known sources with high reputations in Bangladesh. All the sources except sources of personal blogs domain are from reputed organisations and thus can be considered reliable sources of error-free especially spelling mistakes-free source of texts. Moreover, since we have used the automatic crawling and scraping technique

TABLE 5. Distribution of the function words in the corpus.

Group-1		Group-2		Group-3	
Word	%	Word	%	Word	%
ও	1.110	ও	1.203	না	1.198
করে	0.993	না	1.121	করে	1.039
এ	0.923	করে	1.031	ও	0.967
থেকে	0.834	থেকে	0.924	এই	0.912
করা	0.811	করা	0.903	এবং	0.867
না	0.701	এই	0.867	এ	0.802
এই	0.692	হয়	0.817	যে	0.767
হয়	0.667	এবং	0.754	থেকে	0.699
আর	0.632	তার	0.698	তার	0.683
তিনি	0.621	যে	0.634	হয়	0.607

instead of manually typing each word, this automatically gives us the advantage of not manually checking each word for spelling mistakes which could be miserable for such an enormous corpus. However, with our volunteers' help, we manually checked for any inconsistencies from the very beginning of the corpus creation. We are preparing for formal accreditation of the corpus from the Bangla Academy.[3]

G. CORPUS STATISTICS

Before going further with this corpus, we have found different statistical values of it. The percentage of occurrence of each letter in the corpus is demonstrated in Table 6. On the other hand, Table 7 demonstrates the percentage of occurrence of the initial letter of words in the corpus. Finally, the top 40 frequent words in our corpus have been shown in Table 8.

H. BUILDING THE LEXICON

A lexicon is an inventory of lexemes. Lexicon also contains bound morphemes, compound words, and idiomatic expressions. On the other hand, a standard dictionary contains only root words, usually excluding bound morphemes [36]. Hence, a lexicon is a core part of a spell checker. We have created our lexicon by extracting words from our monolingual text corpus, NHMono01. The procedure of creating and filtering our lexicon is explained in Algorithm 1.

At first, we filter out all punctuation from the corpus. Then, we remove the redundant lexemes and keeps only the unique lexemes in the lexicon file. Then we sort the whole lexicon alphabetically to increase the readability. Finally, the lexicon goes through a final lexeme filter.

Our corpus contains many garbage, informal words of different dialects, and names of persons, places, and objects. Our lexicon must not consist of these informal, garbage words. However, separating these words from actual formal words is a challenging task. We have removed these unwanted words based on a semi-automatic approach. We have removed these words based on the frequency of each word and then have manually examined certain words. Since there is no standard threshold available as reference, we have manually studied NHMono01, and we have observed that if a word's frequency is less than 10 in NHMono01, it is usually an unwanted, garbage word such as a name of a person, place, or object. Therefore, we remove it immediately. However, if a word's frequency is between 10 and 50, it is difficult to decide by the machine since, between this range, we have observed a combination of both actual and garbage words. Thus, we check those words manually, and only correct words are included in the final lexicon file. If the frequency of a word is greater than 50, then we consider it as a correct word and store it

TABLE 6. Percentage of occurrence of each letter in the corpus.

letter	%	letter	%	letter	%	letter	%
া	10.557	য়	1.705	থ	0.672	ৃ	0.15
ে	8.768	হ	1.505	ষ	0.668	ঞ	0.104
র	8.384	জ	1.411	ভ	0.625	ৈ	0.07
্	6.13	ট	1.38	অ	0.559	ৌ	0.063
ি	5.477	শ	1.278	খ	0.551	ঢ	0.045
ন	5.421	ৎ	1.226	ড	0.448	ঝ	0.041
ক	4.37	গ	1.204	উ	0.428	ঈ	0.013
ব	3.705	্য	1.089	ং	0.417	ঃ	0.007
ত	3.536	ছ	1.088	ণ	0.383	ঋ	0.006
ল	2.952	ই	1.022	ফ	0.337	ঐ	0.006
স	2.925	আ	0.986	ঁ	0.281	ঊ	0.003
ম	2.866	এ	0.974	ড়	0.238	ঢ়	0.001
প	2.431	ী	0.928	ঠ	0.211	ঔ	0.001
দ	2.214	চ	0.887	ঙ	0.183		
য	2.152	ও	0.794	ঘ	0.168		
ু	1.733	ধ	0.705	ূ	0.157		

TABLE 7. Percentage of occurrence of initial letter of words in the corpus.

letter	%	letter	%	letter	%	letter	%
ব	9.024	জ	3.104	থ	1.32	ঝ	0.094
ক	8.898	অ	3.098	ছ	1.164	ঈ	0.048
স	8.49	র	2.704	ল	1.087	ঐ	0.032
প	7.931	গ	2.448	খ	0.979	ঋ	0.028
ম	5.441	শ	2.405	ফ	0.944	ষ	0.020
আ	5.291	ও	2.059	ট	0.808	ঊ	0.015
হ	4.873	চ	2.027	ঘ	0.654	ঔ	0.003
এ	4.863	য	1.798	ধ	0.545	ি	0.002
ন	4.685	উ	1.571	ড	0.429		
দ	4.496	ভ	1.362	ঢ	0.24		
ত	3.58	ই	1.348	ঠ	0.104		

[3] Bangla Academy is an autonomous institution funded by Bangladesh government to promote and foster the Bangla language, literature, and culture.

TABLE 8. The top 40 frequent words in our corpus.

Word	%	Word	%	Word	%	Word	%
ও	1.218	তিনি	0.407	করেন	0.332	ওই	0.234
করে	0.744	রয়েছে	0.392	এক	0.325	পর	0.223
এ	0.723	মন্তব্য	0.387	একটি	0.313	কিন্তু	0.217
থেকে	0.651	প্রদান	0.374	নিয়ে	0.272	বলে	0.216
না	0.63	ইহাতে	0.366	করতে	0.266	কথা	0.216
করা	0.563	জন্য	0.364	এবং	0.263	প্রথম	0.202
বলেন	0.537	হবে	0.363	মধ্যে	0.242	ছিল	0.199
এই	0.488	হয়েছে	0.36	গতকাল	0.242	দুই	0.199
হয়	0.442	সঙ্গে	0.343	আর	0.237	তবে	0.195
বন্ধ	0.428	তাঁর	0.337	গত	0.236	এর	0.186

Algorithm 1 Creating and Filtering the Lexicon

```
corp ← read the whole corpus file
corp_filt ← filter out all punctuation such as, '।'(full stop),, (comma) etc and any special characters from corp
Lex ← initialize array for lexemes
for each lexeme $L_i$ from corp_filt do
    if $L_i$ does not exists in Lex then
        insert $L_i$ in Lex as newline(\n)
    end if
end for
for each lexeme wordLex from Lex do
    wordFreq ← find word frequency of wordLex in Corpus NHMono01
    (reason behind thresholds below explained in this subsection)
    if wordFreq $< 10$ then
        remove the word wordLex from Lex immediately
    else if wordFreq $>= 10$ and wordFreq $<= 50$ then
        tempStoreArray ← wordLex
    else
        wordLex is a regular frequent valid word. Bring no change.
    end if
end for
check each word from tempStoreArray manually. If any word seems unfit remove it from tempStoreArray
combine Lex and tempStoreArray
sort Lex in alphabetical order (words starting with the character 'অ'(0985) should be at the beginning and 'হ'(09B9) at the last)
write it to the final lexicon file
```

in our main lexicon file directly. The thresholds here are set based on our study on NHMono01, and therefore, these thresholds may not provide desired results in other corpora. Finally, we have distributed the lexicon into several smaller lexicons so that we have to process fewer words at a time instead of the whole lexicon. Distributed lexicons boost the processing speed, reduce the processing time and increase readability. We have distributed the lexicons using a simple hash function that returns only the string length of a given lexeme. The lexicon is consists of over 1 million words, 1,043,106 words to be precise, and the file size is roughly 30 megabytes.

We have also stored the pre-calculated Double Metaphone Values (explained in subsection V-B) with each lexeme in separate files (keeping main lexicon files intact) for the spell checker. At first, we have designed the proposed system to find the Double Metaphone values of lexemes on the fly. However, this method slows down the processing speed drastically. The processing time of our proposed system is reduced significantly by using the pre-calculated Double Metaphone values.

I. DISTRIBUTION OF THE DATASETS

To make the corpus and lexicon easy to process and readily available to all researchers, both the corpus and the lexicon files are stored in the most common file formats, i.e., the `CSV` and `TXT` formats. All individual files and the aggregated single files for both the corpus and the lexicon are stored in the institutional private repository. The file sizes of the whole corpus and the whole lexicon are roughly 2.0GB and 30MB, respectively. Other minor datasets, i.e., Numerical suffix dataset and three test datasets, are also available in the repository. Numerical suffix dataset is a minor dataset that contains a list of all possible suffixes for Bangla numbers. All these datasets are freely available and can be found at https://git.io/JzJ4w.

V. DEVELOPING THE SPELL CHECKER

This section demonstrates different aspects of the proposed spell checker segment, explaining different phonetic algorithms, types of spelling errors, procedures, and algorithm of the proposed spell checker segment.

Phonetic algorithms are essential for an all-inclusive spell checker. Soundex is a phonetic algorithm designed in the 1900s [13], [37]. The Soundex algorithm phonetically encodes a group of similar-sounding consonant characters. The process usually excludes vowels except for the vowels at the beginning of the word. Soundex is an ancient phonetic algorithm and has many imperfections. Metaphone is another phonetic algorithm developed in 1990 by Lawrence Philips [15] that improved the earlier Soundex algorithm in several aspects. Metaphone algorithm is more intricate than the previous algorithms because it included new rules to handle different spelling inconsistencies. In 2000, the same author of Metaphone proposed a revised version of the Metaphone called Double Metaphone [15]. This is a highly sophisticated phonetic algorithm containing all multifaceted rules to grip several types of elocution. Besides, it makes name searching simpler since Double Metaphone considers the pronunciation of different names as well. Double Metaphone provides a primary and a secondary code for a single word.

This gives us advancement over other phonetic algorithms. Since Bangla has numerous words with different pronunciations in different contexts, Double Metaphone gives us the perfect solution.

On the other hand, 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. Equation 4 shows the Levenshtein's distance between two strings a and b, which is specified by $\text{lev}_{a,b}(|a|, |b|)$ where, the length of strings a and b are $|a|$ and $|b|$ respectively.

$$\text{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \text{lev}_{a,b}(i-1,j)+1 \\ \text{lev}_{a,b}(i,j-1)+1 \\ \text{lev}_{a,b}(i-1,j-1)+1 \end{cases} & \text{otherwise.} \end{cases} \tag{4}$$

TABLE 9. Levenshtein's distance between two words.

Lexeme	Edit distance with misspelt word “বিদান(bidāna)”
দান (dāna)	2
বিধান (bidhāna)	1
অনুদান (anudāna)	3

Table 9 shows the Levenshtein's edit distance between the misspelt Bangla word “বিদান” and some similar words from the lexicon. From the above table, we can see that the word “বিধান” has the smallest edit distance with the word “বিদান”. Levenshtein's distance can be used to find the dissimilarities between two Double Metaphone encoded strings. Table 10 illustrates Levenshtein's distance between two Double Metaphone encoded words. For example, our Double Metaphone encoding technique returns “bdn” for the misspelt word “বিদান”. Since the Double Metaphone encoding distance between the words “বিদান” and “বিধান” is 0, our system assumes the correct word for the misspelt word “বিদান” should be “বিধান”.

TABLE 10. Levenshtein's distance between two double metaphone encoded words.

Lexicon word	Double Metaphone of lexicon word	Levenshteins distance with misspelt double metaphone encoded word bdn
দান	dn	1
বিধান	bdn	0
অনুদান	ndn	1

A. TYPES OF SPELLING ERRORS

According to Ahmed [38], Bhatti *et al.* [39], and Dong *et al.* [40] spelling mistakes can be divided into following types. Our proposed system is capable of detecting all these different types of spelling mistakes.

1) TYPOGRAPHIC ERROR

Typographic errors occur due to typing mistakes. For example, we usually place our fingers on a 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. Some examples of typographic errors have been shown in Table 11.

TABLE 11. Typographic error example in Bangla.

Error Type	Misspelt Word	Correct Word	Transliterations (misspelt & correct)
Insertion	দেইশ	দেশ	dē'iśa & dēśa
Deletion	বাংল	বাংলা	bānla & bānlā
Substitution	বাংলাদেস	বাংলাদেশ	bānlādēsa& bānlādēśa
Transposition	কমল	কলম	kamala & kalama

2) COGNITIVE ERROR

A cognitive error occurs due to the similar pronunciation of alphabets. Cognitive error is also called Phonetic error. For example, in Bangla, alphabets ‘প’(PA) and ‘ফ’(PHA) sound the same. In Bangla, there are many other alphabets with similar sound (e.g. ‘গ’(GA)-‘ঘ’(GHA), ‘শ’(SHA)-‘স’(SA)-‘ষ’(SSA), ‘ব’(BA)-‘ভ’(BHA) and so forth). That is why cognitive error is widespread in Bangla. Table 12 shows some examples of cognitive error in Bangla.

TABLE 12. Cognitive error example in Bangla.

Misspelt Word	Correct Word
দেস (dēsa) or দেষ (dēṣa)	দেশ (dēśa)
বাষা (bāṣā) or ভাশা (bhāśā)	ভাষা (bhāṣā)
গোলাফ (gōlāpha) or ঘোলাপ (ghōlāpa)	গোলাপ (gōlāpa)

3) VISUAL ERROR

A visual error occurs due to the same visual shape of alphabets. Our study shows that in Bangla, we have some similar-looking alphabets as well. Table 13 shows the alphabets that look similar in Bangla.

These similar alphabets may lead to visual errors in Bangla words. Table 14 shows some examples of errors in Bangla words that have been originated from visual errors.

4) RUN-ON ERROR

This type of error occurs when two correct words have been joined since no inter-word gap is given by mistake. Table 15 shows an example of missing space/ run-on error. This type of error will be detected easily since joining two correct words might create an incorrect word that should not exist in our lexicon or anywhere in the corpus.

5) SPLIT-WORD ERROR

This type of error occurs when one word is split into two strings of characters because a gap is mistakenly inserted

TABLE 13. Visually similar Bangla alphabets.

উ (U)	=	ঊ (UU)
ব (BA)	=	র (RA)
ন (NA)	=	ণ (NNA)
ড (DDA)	=	ড় (RRA)
ঢ (DDHA)	=	ঢ় (RRHA)
য (YA)	=	য় (YYA)
য (YA)	=	ষ (SSA)

TABLE 14. Visual error example in Bangla.

misspelt Word	Correct Word
পাহাড (pāhāḍa)	পাহাড় (pāhāṛa)
বিশেয (biśēṣa)	বিশেষ (biśēṣa)
ফুটরল (phuṭarala)	ফুটবল (phuṭabala)

TABLE 15. Run-on error example in Bangla.

Misspelt Word	Correct Word
মিষ্টিআলু (miṣṭi'ālu)	মিষ্টি আলু (miṣṭi ālu)

within the word. Our spell checker may or may not detect this error as a misspelt word(s) since these individual word fragments may or may not meaningful in Bangla. If an individual string is incorrect, only then that specified string will be marked as an error. Table 16 shows an example of split-word error.

TABLE 16. Split-word error example in Bangla.

Misspelt Word	Correct Word
মোটর গাড়ি (mōṭara gāṛi)	মোটরগাড়ি (mōṭaragāṛi)

All the above errors can be summarised into two types of errors, i.e., Non-word error and Real-word error.

6) NON-WORD ERROR

Non-word error results from a spelling error where the word itself is not in the lexicon/dictionary and is not known. For example, mistakenly spelling 'কমলা (kamalā)' into 'কমমা(kamamā)' is a non-word error since 'কমমা' is not in our lexicon.

7) REAL-WORD ERROR

Real-word error is due to misspelling a word to make another word that is in the lexicon. For example, mistakenly spelling 'কমলা(kamalā/Orange)' into 'কোমল(kōmala/Soft)'. This is a real-word error because 'কোমল' is in our lexicon; however, this is wrong in this context. Our proposed method does not consider it as a spelling error. Instead, it considers the error as a semantic error [41] and shows it as a grammatical error.

B. PROPOSED DOUBLE METAPHONE ENCODING

We have used Double Metaphone encoding as our phonetic algorithm to detect phonetically similar Bangla words. The two different codes (i.e., a primary and a secondary) for a single word provided by the Double Metaphone help to identify different pronunciations of the same alphabet. Our encoding table is based on the encoding table projected by Uzzaman and Khan [3], which is the only Double Metaphone encoding available in Bangla. We have decided to use their encoding table after a rigorous study by manually checking each Bangla alphabet, their IPA (International Phonetic Alphabet) symbols, and the standard pronunciation of each character regulated by Bangla Academy. We have proposed the following minor improvements for the encoding table:

- **Letter 'ঋ' (U+098B)**: We code the letter 'ঋ' as 'r' instead of 'ri' in our encoding table. For a proper suggestion, the edit distance should be as less as possible where 0 is considered the closest. We compare both codes ('ri' and 'r') side by side to observe the difference. Table 17 demonstrates an example where the edit distance of the 'ri' code is 1. On the other hand, code 'r' results in an edit distance of 0. If we code 'ঋ' as 'r', the spell checker shows 'ঋষি(r̥ṣi)' at the top of the suggestion list for the misspelt word 'রিশি(riśi)' whereas, for the code 'ri', the system does not show 'ঋষি' in the suggestion list since it has higher edit distance value.

TABLE 17. Example comparison of 'ঋ' between previous and our approach.

Approach	Word	Primary Code	Edit Distance
'ঋ' coded ri [3]	ঋষি(correct) রিশি(incorrect)	ris rs	ED(ris|rs) = 1
'ঋ' coded r (Our Approach)	ঋষি(correct) রিশি(incorrect)	rs rs	ED(rs|rs) = 0

- **Letter '◌ৃ' (U+09C3)**: We have observed that there is no mention of the letter '◌ৃ' in [3] separately. The authors of that paper gave an example that demonstrates that they considered the letter '◌ৃ' same as the letter 'ঋ'. However, these two letters are different with different Unicode codes, U+098B and U+09C3, respectively. Table 18 shows an example of this scenario, where we have demonstrated the consequences of not coding the letter '◌ৃ' uniquely and separately. Without a unique code, the letter provides an edit distance of 1, whereas our approach (coding the letter with 'r') provides an edit distance of 0. Thus, unlike the previous approach (not assigning any code for '◌ৃ'), our coding successfully shows 'বিকৃত (bikr̥ta)' at the top of the suggestion list for the incorrect word 'বিকরিত (bikarita)'.
- **Letter 'ঢ়'**: In the previous encoding [3], the letter 'ঢ়' (Rha) is incorrectly mapped to the Unicode code point 'U+09A2', which is actually the Unicode code point of the letter 'ঢ' (Ddha) that might lead to incorrect decisions by a system. Although both the letters look similar,

TABLE 18. Comparison of coding '◌ৃ' between previous and our approach.

Approach	Word	Primary Code	Edit Distance
'◌ৃ' not coded [3]	বিকৃত (correct)	bkt	ED(bkt|bkrt)=1
	বিকরিত (incorrect)	bkrt	
'◌ৃ' coded with r (Our Approach)	বিকৃত (correct)	bkrt	ED(bkrt|bkrt)=0
	বিকরিত (incorrect)	bkrt	

the pronunciation and meaning of these two letters are different. Therefore, we have changed the mapping to the correct name and Unicode as shown in Table 19.

TABLE 19. Unicode correction for letter 'ঢ়'.

Approach	Letter	Unicode
Previous Encoding [3]	'ঢ়'	U+09A2
Our Approach	'ঢ়'	U+09DD

We have made a few other negligible changes and furnished the encoding table before using it in our proposed approach.

C. METHODOLOGY OF THE SPELL CHECKER

The proposed algorithm for our spell checker system is shown in algorithm 2. The spell checker system removes punctuation from any given texts and splits them using whitespace, creating smaller chunks called lexemes. Since the proposed spell checker is based on the Bangla language, it avoids processing any English word or number (based on the Unicode code point[4]), and considers it as a correct lexeme. Validating a Bangla number is more straightforward. If a lexeme is entirely numerical, containing all Bangla digits (such as ৪৫), the system marks it as a correct lexeme. However, if the lexeme is alphanumerical (such as ৪৫টি), the suffix is then split from the numerical part. The spell checker marks it as the correct lexeme if the suffix exists in the suffix dataset; otherwise, it is identified as incorrect (such as ৪৫টঁ). A list of lexemes is suggested based on the suffix dataset embedded with the numerical part of the erroneous lexeme.

On the other hand, for regular Bangla lexemes, the spell checker generates a hash key against each lexeme. The key is the string length of the lexeme. The system then searches for that lexeme in the lexicon using the hash-key (K). The system considers a lexeme valid if it finds the lexeme in the lexicon. Otherwise, it marks the lexeme as misspelt. To give appropriate suggestions for a misspelt lexeme, the system generates the DMV (Double Metaphone Values) of the lexeme and finds the edit distance between the DMV and the pre-calculated DMVs of all lexemes in the lexicon-K (lexicon of string length K) along with lexicon-($K-1$) (lexicon of a string length $K-1$), and lexicon-($K+1$) (lexicon of a string length $K+1$). Following this method, the spell checker generates a suggestion list of lexemes where the edit distance is equal or ±1. The system then ranks all the lexemes from this suggestion list based on a summation of different weights (the lower the weight, the better/closer to the misspelt lexeme), and subtracts the summation result from 100. Finally, it displays the highest 5 lexemes in the final suggestion list. The summation of weights includes three measurements. First, in which lexicon a lexeme from the tentative suggestion list belongs to. For example, if a suggestion lexeme belongs to lexicon-K, which might be a good suggestion since the string length of misspelt lexeme is also K, we assign less weight (the less, the better) to it than the lexemes that belong to the other two lexicons $K-1$ and $K+1$. Second, the edit distance is based on the DMV values between the misspelt lexeme and each lexeme in the tentative suggestion list. Lastly, the edit distance is based on direct Bangla characters of the lexeme between the misspelt lexeme and each lexeme in the tentative suggestion list.

A typographic error is a spelling mistake that does not follow linguistic rules due to a mistake in typing the word. Similarly, a cognitive error occurs due to similar pronunciation, and a visual error occurs due to similar looks of Bangla alphabets. Furthermore, a run-on error occurs due to the removal of space between two words. The erroneous words generated by these spelling mistakes do not exist in any of our distributed lexicons with their DMV values. Thus, the spell checker can detect the above-mentioned spelling errors accurately and can suggest appropriate words. On the other hand, split-word errors or any errors that may create real-word errors are detected through the grammar checker's language model probability. We use the combination of Bigram and Trigram to generate a context analysis using the frequency analysis between words and thus, identify a real-word error.

VI. GRAMMAR CHECKER

In this section, we have presented the methodology of the grammar checker. Building a full-fledged and generalised Bangla grammatical error correction approach is really a challenging task due to the complicated grammatical rules of the Bangla language, especially if the text is mixed with spelling errors.

Algorithm 3 explains the procedure of the grammar checker in detail. At first, it splits the input Bangla text into sentences using '।' (Bangla full-stop), and then each sentence goes through a basic grammatical error check. The basic check includes identifying and suggesting a solution for redundant words error, additional whitespace errors, and punctuation errors. The proposed grammar checker is based on the N-gram (to be specific, bigram and trigram) language model probability. An N-gram is a contiguous sequence of N items from a given sample of texts. In our case, N is equal to either 2 (bigram) or 3 (trigram) based on the size of a given sentence. Bigram is used for short sentences (where the word count of a sentence ≤ 3), and trigram is for long sentences (where the word count of a sentence > 3).

[4] https://www.unicode.org/charts/

Algorithm 2 Spell Checker

```
IN ← input text
IN ← remove punctuations from IN (will be added back later after processing)
INarray ← split IN into separate lexemes using whitespace
for each lexeme $L_i$ in INarray do
    if $L_i$ is English word or number then
        mark $L_i$ as English word/number and avoid processing.
    else if $L_i$ is Bangla number then
        if $L_i$ is entirely numerical then
            mark $L_i$ as correct lexeme
        else
            LiSuf ← split suffix from $L_i$
            if LiSuf exists in our suffix dataset then
                mark $L_i$ as correct lexeme
            else
                mark $L_i$ as incorrect
                show suggestions from the suffix dataset embedded with numerical part
            end if
        end if
    else
        K ← generate string length of $L_i$ as hash key
        if $L_i$ exists in lexicon-K then
            mark $L_i$ as correct lexeme
        else
            mark $L_i$ as incorrect or misspelt lexeme
            DMV ← generate Double Metaphone values of $L_i$
            ED ← calculate levenshtein's edit distance of DMV with all pre-generated DMV of lexemes in lexicon-K, lexicon-$K - 1$ and lexicon-$K + 1$
            SUG ← store all the lexemes where $ED <= 1$
            for each lexeme SUGlex in SUG do
                SUM ← 0
                if SUGlex belongs to lexicon-K then
                    SUM ← SUM + 1
                else if SUGlex belongs to lexicon-$K - 1$ or $K + 1$ then
                    SUM ← SUM + 2
                end if
                EDDMV ← generate edit distance between DMV value of $L_i$ and DMV value of SUGlex
                EDWord ← generate edit distance between $L_i$ and SUGlex (text to text)
                SUM ← SUM + EDWord + EDDMV
                RANKSUGlex ← 100 − SUM
                RANK ← Store SUGlex and RANKSUGlex
            end for
            present top 5 lexemes from RANK (if available)
        end if
    end if
end for
```

Our study finds that bigram probability is not efficient enough to detect grammatically incorrect sentences if a sentence is long or complex in structure. Trigram language model probability increases the accuracy in almost all sentence structures, which has the lowest perplexity amongst unigram, bigram, and trigram [42]. Equation 5 shows the formula of an N-gram and equation 6 shows the formula of estimating the probabilities where w represents word, C represents count, and N represents N of N-gram (sequence of N items). If the language model probability is above a threshold, we consider the sentence as grammatically correct otherwise incorrect. The proposed system shows appropriate suggestions for the incorrect sentence. Although it is rare in such a large corpus like the proposed corpus, the system uses Laplace smoothing

Algorithm 3 Grammar Checker

```
$IN \leftarrow$ input text
$SEN \leftarrow$ split $IN$ into separate sentences using ।(bangla full-stop)
for each sentence $SE_i$ in $SEN$ do
    [Start Basic Check]
    if $SE_i$ contains any redundant words consecutively then
        Mark it as grammatical error and suggest to remove redundant word.
    end if
    if $SE_i$ contains any additional whitespaces in between words then
        Mark it as grammatical error and suggest to remove whitespace(s).
    end if
    if basic punctuation error found(such as redundant punctuation or missing punctuation) then
        Mark it as grammatical error and suggest to remove or add punctuation
    end if
    [End Basic Check]
    $LEN_i \leftarrow$ generate word count of $SE_i$
    if $LEN_i <= 3$ then
        $PROB_i \leftarrow$ find language model probability of $SE_i$ based on bigram counts
    else
        $PROB_i \leftarrow$ find language model probability of $SE_i$ based on trigram counts
    end if
    if $PROB_i$ is above a threshold (in our case 0) then
        Mark $SE_i$ as grammatically correct
    else
        Mark $SE_i$ as grammatically incorrect
        Find Cosine Similarity(CS) between $SE_i$ and sentences in corpus
        Keep records of results in a temporary file for sentences similar to $SE_i$
        Display top two closest match as a potential grammatical correct sentence for $SE_i$
    end if
end for
```

add 1 policy [42] to deal with zero probability.

$$P(w_1^n) = \prod_{k=1}^{n} P(w_k | w_{k-N+1}^{k-1}) \tag{5}$$

$$P(w_n | w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1} w_n)}{C(w_{n-N+1}^{n-1})} \tag{6}$$

On the other hand, many existing Bangla grammar checkers and parsers detect errors without suggesting any solution for the grammatical mistakes. Suggesting a solution helps a user to correct the problem immediately and thus reduces the time of correcting grammatical mistakes. In order to provide appropriate suggestions, the proposed system uses the cosine similarity algorithm [43] to measure the similarity between a grammatically incorrect sentence and the sentences already stored in the corpus. Cosine similarity is a metric used to measure how similar two documents are, irrespective of their size. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space. The smaller the angle, the higher the cosine similarity. Bangla characters are double-byte unlike English, and thus, Bangla sentences take more time in processing. After studying several similarity measurement algorithms and different aspects of the Bangla language, the cosine similarity algorithm has been used to find similarities between a given sentence and the sentences in the corpus. Cosine similarity provides a better accuracy rate and lowers the processing time as well. In our case, it takes less than 1 second on average to process 100 MB of double-byte Bangla sentences. Equation 7 shows the formula of the cosine similarity algorithm where θ represents the angle of the cosine, and A and B are two vectors.

$$cos(\theta) = \frac{A.B}{\|A\| \, \|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{7}$$

VII. EXPERIMENTAL SETUP, RESULTS, ANALYSIS, AND IMPLICATIONS

In this section, we have demonstrated and analysed different experimental results of both spell and grammar checkers. For ease of understanding, the experimental results of spell and grammar checkers have been shown in separate subsections.

A. EXPERIMENTAL SETUP

The experiment has been carried out in a high-performance computer with 32GB RAM, 6GB Graphics Card, and Intel Xeon Processor. Python 3.6 is the core language in the experiment with the scikit-learn library [32]. The proposed system has been tested with three different test datasets, which contain different types of errors to evaluate the performance of the proposed system. Table 20 shows the size of all the test datasets. In order to create the test sets, we have scraped new sentences from the same sources mentioned in Table 3 and intentionally, manually manipulated sentences by putting specific errors according to our needs. Before manipulating the sentences, a copy of the datasets is stored to verify our system output later and determine the accuracy rate.

TABLE 20. **Test dataset size.**

Data Set Type	Total sentences	Total words
Test-dataset-spell	8042	37377
Test-dataset-grammar	7545	34255
Test-dataset-both	8835	41236

In our datasets, `Test-dataset-spell` contains sentences with misspelt word(s), `Test-dataset-grammar` contains sentences with grammatical mistake(s), and `Test-dataset-both` contains both spelling and grammatical mistakes in each sentence.

B. RESULTS AND PERFORMANCE ANALYSIS

Table 21 shows some examples of error detection for different types of spelling errors. It also shows the suggestions for each error generated by the proposed spell checker. The position where the error has occurred is indicated in red colour.

Example 1 has no spelling mistakes, and therefore, no suggestion is generated. Two spelling mistakes (a cognitive and a visual error) have been identified for example 2, and for each case, several suggestions have been provided. A numerical error and its tentative suggestions are shown for example 3. The word 'Amazon' in example 4 has no impact on the spell checker since it avoids processing foreign words (i.e., English). Finally, example 5 has three errors (a cognitive, a visual, a run-on error) in a single sentence with corresponding suggestions.

Table 22 shows some examples of error detection for different types of grammatical errors. The position where the error has occurred is indicated in red colour. A word redundancy error, an additional whitespace error, and two punctuation errors are shown in examples 1, 2, and 3, respectively, with suggestions to fix these errors. Examples 4 and 5 demonstrate a subject-verb agreement error and a complete sentence structure error while suggestions have been generated for both cases. Table 23 demonstrates some sample example cases where the proposed system fails to detect errors or suggest appropriately. The explanations of the reasons for failures and how these failures can be overcome are also mentioned in the table.

For measuring the accuracy of our system, we have tested our system with the sentences mentioned in Table 20 and verified the results with the correct version of the sentences. If our system detects all the errors in a sentence and provides the correct suggestions as well, only then we consider it as a 'success'. On the other hand, in some sentences, the system successfully detects and suggests one error but fails to detect all the errors in the sentence, especially in the sentences containing both spelling and grammatical errors. This is a 'failure'. In addition to that, when the proposed system detects all the errors successfully but does not provide the appropriate suggestions is also a 'failure'. The test summary is illustrated in Table 24. In this table, accuracy has been measured using the following equation:

$$Accuracy = \frac{CDS}{TS} * 100 \tag{8}$$

where, $CDS =$ The number of sentences where all spelling or grammatical errors are successfully detected and successfully suggested with the appropriate solution. $TS =$ The number of sentences in that particular test dataset.

The accuracy rate is the highest when we test our system with the test dataset that contains spelling errors only and the lowest when we test our system with the test dataset that contains both spelling and grammatical errors. The average accuracy rate is roughly 94.88%. Although it is rare, it fails to place the closest word at the top of the suggestion list in few cases of the spell checker. Instead, it places the closest word at the bottom of the suggestion list. This happens because similar-sounding Bangla alphabets misguide the proposed system in this rare situation. The reason behind the grammatical error detection failure is the statistical approach that lacks similar sentences in the corpus.

C. COMPARISON WITH THE EXISTING SOLUTIONS

Lexicon-based spell checkers need substantially large data to cover, as many words as possible, to detect errors, and to generate the appropriate suggestions. In terms of the size of the lexicon, our proposed spell checker shows major advancement over the existing solutions. Note that [19] is based on a lexicon with the size of 50,000 words, where our proposed lexicon consists of over 1 million words. On the other hand, [15] and [17] did not mention about their lexicon sizes and collection procedures. However, they tested their spell checker with datasets containing 1,607 and 2,450 words, respectively. Most of these approaches, including recent work [20] and [21] handle specifically typographic errors, [22] handles particularly real-word errors where the proposed approach handles all different types of errors that exist in Bangla language. Another significant difference between our proposed spell checker and [13] to [22] (excepts [20]) is that they tested their spell checker on test sets consisting of only single words instead of complete sentences, which might not reflect the real-life situations and complexities. Finally, all the previous approaches, including the recent recurrent neural network (RNN)-based spell checker (82.86%) [22] show lower accuracy rates than the proposed spell checker, which is 97.21%.

When a statistical approach is being used, the more data a grammar checker gets, the better performance it can provide. Our proposed grammar checker leverages its performance using our very large-sized corpus. [24] to [31] (except [26], [29], and [30]) proposed some CFG based

TABLE 21. Spell checker examples.

SL	Input	Results	Suggestion
1	বাংলা সাহিত্যের সমসাময়িক লেখকদের সামগ্রিক অবদান চিহ্নিত করে তাঁদের প্রতিভাকে স্বীকৃতি দেওয়াই বাংলা একাডেমির সাহিত্য পুরস্কার প্রদানের উদ্দেশ্য।	[No spelling error]	*No suggestion*
2	প্রতি বচর আয়োজিত একুশে গ্রন্থমেলা ও অনুষ্ঠানমালার উদ্বোধন অনুষ্ঠানে মাননীয় অতিথি পুরস্কার প্রাপ্তদের পুবস্কারের অর্থমূল্যের চেক ও স্মারক প্রদান করেন।	[2 spelling errors found] প্রতি বচর[1] আয়োজিত একুশে গ্রন্থমেলা ও অনুষ্ঠানমালার উদ্বোধন অনুষ্ঠানে মাননীয় অতিথি পুরস্কার প্রাপ্তদের পুবস্কারের[2] অর্থমূল্যের চেক ও স্মারক প্রদান করেন।	1. বচরঃ বছর, বছরও, বছরে, বছরই, বছরটা 2. পুবস্কারেরঃ পুরস্কারের, পুরস্কারে, পুরস্কারেই, পুরস্কারেরই, পুরস্কারেরও
3	এই বনে বড় উপজাতি গোষ্ঠীর সংখ্যা মোট ২২ট বা আরও কম। মোট ক্ষুদ্র নৃতাত্ত্বিক গোষ্ঠীর সংখ্যা ১০ লাখেরও বেশি।	[1 spelling error found] এই বনে বড় উপজাতি গোষ্ঠীর সংখ্যা মোট ২২ট[1] বা আরও কম। মোট ক্ষুদ্র নৃতাত্ত্বিক গোষ্ঠীর সংখ্যা ১০ লাখেরও বেশি।	1. ২২টঃ ২২টি, ২২টা
4	পার্শ্ববর্তী দেশ ভারতেও Amazon আছে, কিন্তু বাংলাদেশে নেই।	[No spelling error]	No suggestion
5	বাংলাদেশের দক্ষিণ-পূর্বাঞ্চলে অবস্থিত পাহাডি উপত্যকায় চাষ হয় ঘোলাপ এবং ঢালে বাণিজ্যিকভাবে চাষাবাদের জন্য ভালো মিষ্টিআলু, কলা, আনারস, মাল্টা ও কমলা।	[3 spelling errors found] বাংলাদেশের দক্ষিণ-পূর্বাঞ্চলে অবস্থিত পাহাডি[1] উপত্যকায় চাষ হয় ঘোলাপ[2] এবং ঢালে বাণিজ্যিকভাবে চাষাবাদের জন্য ভালো মিষ্টিআলু[3], কলা, আনারস, মাল্টা ও কমলা।	1. পাহাডিঃ পাহাড়ি, পাহাড়, পাহাড়ে, পাহাড়ের, পাহাড়েও 2. ঘোলাপঃ গোলাপ, গোলাপই, গোলাপও, গোলাপকে, গোলাপে 3. মিষ্টিআলুঃ মিষ্টি আলু

TABLE 22. Grammar checker examples.

SL	Input	Results	Suggestion
1	সব রকমের ফুল তুলনামূলকভাবে ঠান্ডা আবহাওয়া আবহাওয়া এবং রৌদ্রজ্জল জায়গা পছন্দ করে।	[1 grammatical error found] সব রকমের ফুল তুলনামূলকভাবে ঠান্ডা আবহাওয়া আবহাওয়া[1] এবং রৌদ্রজ্জল জায়গা পছন্দ করে।	1. redundancy error: duplicate “আবহাওয়া”
2	এই প্রজন্মের অনেকেও আপনার পূর্ব পুরুষ এবং আপনাকে চিনলেও আপনার পরিবার সম্পর্কে খুব একটা জানে না।	[1 grammatical error found] এই প্রজন্মের অনেকেও আপনার পূর্ব পুরুষ[1] এবং আপনাকে চিনলেও আপনার পরিবার সম্পর্কে খুব একটা জানে না।	1. whitespace error: extra whitespaces
3	ঠিক যখন সেখানে সিংহ তার হাড় হিম করা ডাক দিল,, তখনই আমার জানালার বাইরে শেয়াল ডেকে উঠলো	[2 grammatical errors found] ঠিক যখন সেখানে সিংহ তার হাড় হিম করা ডাক দিল,,[1] তখনই আমার জানালার বাইরে শেয়াল ডেকে উঠলো[2]	1. punctuation error: extra ‘,’ 2. punctuation error: missing end-marker ‘।’
4	তাঁরা এমন প্রযুক্তি ব্যবহার করছে, যা আইন ভঙ্গ করলে স্বয়ংক্রিয়ভাবে ধরা পড়ে। আমরা একই প্রযুক্তি ব্যবহার করতে পারেন।	[1 grammatical error found] তাঁরা এমন প্রযুক্তি ব্যবহার করছে, যা আইন ভঙ্গ করলে স্বয়ংক্রিয়ভাবে ধরা পড়ে। আমরা একই প্রযুক্তি ব্যবহার করতে পারেন[1] ।	1. agreement error: use “পারি”, “পারতাম”
5	সাহায্য করেছিল পশ্চিমারা ধনতন্ত্রের বিরোধগুলোকে বৈধতা দিতে। প্রায় বিশ বছর আগে তাদের পতন ঘটে।	[1 grammatical error found] সাহায্য করেছিল পশ্চিমারা ধনতন্ত্রের বিরোধগুলোকে বৈধতা দিতে।[1] প্রায় বিশ বছর আগে তাদের পতন ঘটে।	1. structure error: use “পশ্চিমারা ধনতন্ত্রের বিরোধগুলোকে বৈধতা দিতে সাহায্য করেছিল।”

parsers for Bangla sentences with reasonably small-sized datasets. References [23], [26], [29], and [30] are moderate-sized statistical grammar checkers where [23] has a corpus of 5,000 words, and [29] has a corpus of 250K sentences collected from two websites, and [30] has two datasets combining 158487 sentences in total. Although authors of [26] have mentioned using a moderately large size corpus of 10 million words, they did not mention how and from where they collected their data or how they processed their corpus. Therefore, they failed to establish the credibility of the corpus. On the other hand, our proposed grammar checker has a corpus of over 100 million words. The accuracy rates of [23], [26], and [30] are 38%, 82%, and 74.5%, respectively. Moreover, the Deep Neural Network-based sequence to sequence learning grammar checker [29] shows an accuracy rate of 79%. The accuracy rates of these existing solutions are comparatively low, while the proposed grammar checker shows a satisfactory accuracy rate of 94.29%.

Finally, all the existing solutions target to solve a partial problem, i.e., either a spell checker or a grammar checker but not both. We are the first to take a holistic approach and develop a spell and grammar checker in a single system. Our combined spell and grammar checker shows a satisfactory accuracy rate of 93.13% with the specific test dataset to test the combined spell and grammar checker. However, the average accuracy rate combining results from the 3 distinct test datasets by the proposed spell and grammar checker is 94.88%. Table 25 shows a summarised comparison between recent and notable spell and grammar checkers with the proposed system.

TABLE 23. Sample examples of failed cases.

SL	Input	Problem/Error	Reason
1	যুক্তরাষ্ট্রে Gugol সবচেয়ে বেশি বেতন দিয়ে থাকে।	Although spelling of the word ‘Google’ is incorrect, the spell checker does not detect it as incorrect.	The entire work is for Bangla language and thus, the spell checker avoids processing any foreign words for correctness.
2	প্রাপ্তবয়স্ক দেশি জাতের গোরু সাধারণত ১৩০ থেকে ১৫০ কেজি ওজনের হয়ে থাকে।	প্রাপ্তবয়স্ক দেশি জাতের গোরু সাধারণত ১৩০ থেকে ১৫০ কেজি ওজনের হয়ে থাকে। The well-known spelling of cow in Bangla is ‘গরু’. However, Bangla Academy recently updated the spelling as ‘গোরু’.	In recent days, Bangla Academy changes spellings of many frequently used Bangla words. Since the changes made in recent days and the proposed lexicon is based on previous public data and contains only high-frequency words, it does not recognise গোরু as a valid word. We are working to introduce these new spellings into the lexicon.
3	প্রতিদিন সকালে গরম জলে কালো নুন মিশিয়ে খেয়ে নিন শরীর সুস্থ থাকবে।	প্রতিদিন সকালে গরম জলে কালো নুন মিশিয়ে খেয়ে নিন শরীর সুস্থ থাকবে। Although the synonym of salt is both ‘লবণ’ and ‘নুন’ in Bangla, the spell checker detects ‘নুন’ as incorrect.	Although Bangla is spoken in both Bangladesh and West Bengal of India, the pronunciation and use of many Bangla words are different in these two places. For example, the words “লবণ, মরিচ, গোসল, আমন্ত্রণ” are commonly used in Bangladesh while the same words are being replaced by “নুন, লঙ্কা, চান, নেমন্তন্ন” in West Bengal, India. Since the sources of the corpus are all from Bangladesh, the spell checker does not recognise many spellings from the West Bengal side. We can solve this issue by adding sources from West Bengal in the upcoming version of the corpus.
4	এই দম্পতির পাঁচ ছেলে ও দুই মেয়ের মধ্যে রফিক ছিলেন বড় সন্তান।	এই দম্পতির পাঁচ ছেলে ও দুই মেয়ের মধ্যে রফিক ছিলেন বড় সন্তান। The person’s name, ‘রফিক’ is recognised as a mistake.	The proposed system fails to identify name entities and recognises name entities as errors due to the absence of Named-entity Recognition (NER) in the system. Lack of a highly accurate Bangla NER necessitated us to avoid using the existing NERs. We are working on to build an NER for the proposed system that will be embedded into the future version.
5	আমার জ্বর জ্বর লাগছে। তুমি দিন দিন রোগা হয়ে যাচ্ছ।	আমার জ্বর জ্বর লাগছে। তুমি দিন দিন রোগা হয়ে যাচ্ছ। The duplicate words in the above example are recognised as incorrect but these duplicate words are correct in this context.	In Bangla language, there is a term ‘দ্বিরুক্ত শব্দ’ that is repeated words with specific meaning. The repeated words usually are adjectives and indicate excess or lack of something. However, the proposed system considers these দ্বিরুক্ত শব্দ as duplicate words error. We need a separate database that will contain a collection of these repeated words with definite meaning.
6	গ্রামে যানজট, কোলাহল, ধোঁয়া ও ধুলা কিছুই নেই। খাবার খেলাম আমরা মাংস দিয়ে।	গ্রামে যানজট, কোলাহল, ধোঁয়া ও ধুলা কিছুই নেই। খাবার খেলাম আমরা মাংস দিয়ে। The structure of the second sentence is grammatically incorrect. The system suggests two sentences: 1) আমরা মাংস দিয়ে খাবার খেলাম। , 2) আমরা মাংস দিয়ে শিকার করি। . The first suggestion is grammatically correct, meaningful and represents the intended meaning of the original sentence. However, the second suggestion, although a grammatically correct one, has a different meaning than the original sentence.	The proposed system shows an incorrect suggestion for the grammatically incorrect sentence due to the text scarcity of similar sentences in the corpus. The text similarity measure algorithm finds that sentence (i.e., the incorrect suggestion) closer to the original sentence than other sentences in the corpus. By increasing the corpus size from various sources will solve this problem.

TABLE 24. Performance summary of the proposed system.

	Spell Checker	Grammar Checker	Spell and Grammar Checker
Dataset used:	Test-dataset-spell	Test-dataset-grammar	Test-dataset-both
Total Sentences:	8042	7545	8835
Error detected:	7818	7114	8228
Error not detected:	257	431	607
Accuracy rate:	97.21%	94.29%	93.13%

TABLE 25. Comparison between previous works and the proposed system.

Approach	Spell/ Grammar	Corpus Size	Lexicon Size	Accuracy
[19]	Spell	-	50,000	96.48%
[21]	Spell	-	959,232	97%
[22]	Spell	-	1,008,759	82.86%
[26]	Grammar	10,000,000	-	82%
[29]	Grammar	1,678,970	-	79%
[30]	Grammar	1,061,863	-	74.5%
Proposed	Both	100,142,522	1,043,106	97.21% & 94.29%

D. IMPLICATIONS

The findings of this study also have important theoretical and practical implications. First, in the field of statistical linguistic research, a reliable, sufficiently large, and balanced corpus is an essential aspect for linguistically complex languages such as Bangla. The lack of such corpus in Bangla is a bottleneck for the advancement of the linguistic research that is expected to be solved by the proposed NHMono01 corpus. Second, the proposed lexicon is a generous inventory of Bangla lexemes that could be used by the future researchers in numerous areas of Natural Language Processing (NLP), such as spell checker, sentiment analysis, opinion mining, sentence completion, and other linguistic research areas. Third, several related research works, including the very recent works on the spell and grammar checker in Bangla language have been explored and analysed in this paper. Moreover, different types of errors have been shown, with examples in the Bangla language. Future researchers on similar topics may narrow down and set precise research goals based on this information in Bangla. Fourth, the general approach of the proposed spell and grammar checker can be applied to other similar multi-byte languages that share the same linguistic rules and features. Thus, researchers from other similar languages would find significant guidance to develop their own from the proposed spell and grammar checker. On the other hand, we are expecting that our proposed spell and grammar checker would bring a positive impact on publishers, bloggers, and writers of the Bangla language.

VIII. CONCLUSION AND FUTURE WORK

The proposed system exhibits a Bangla spell and grammar checker. It also demonstrates the development procedure of the largest Bangla monolingual corpus. By extracting unique lexemes from this corpus the largest Bangla monolingual lexicon has also been built. The paper illustrates every step of building the corpus, lexicon, and spell and grammar checker. The proposed work achieves the accuracy rate of 97.21% in the spell checker segment, 94.29% in the grammar checker segment, and 93.13% in the combined spell and grammar checker. The approach can be used for any other low-resource languages that have similar structure to Bangla. Moreover, the lack of adequate corpus and lexicon is one of the major hindrances that slowed down the progress of NLP researchers in Bangla language. Since the corpus and lexicon are openly accessible, both researchers from academia and developers from industries in this area will be benefited.

In the future, authors are planning to develop an open-source mobile app and an extension to browsers for users to use the proposed Bangla spell and grammar checker. The system would collect users' usage data with users' permission, e.g., selection of suggestions for an erroneous word or sentence. This data would be used to improve the next version of the proposed spell and grammar checker application.

ACKNOWLEDGMENT

The authors would like to thank all the faculty and staff members of the Computer Science and Engineering Department of United International University, Bangladesh, who supported the work by providing software and hardware resources.

REFERENCES

[1] M. Jishan, K. R. Mahmud, A. Azad, M. Rashid, B. Paul, and M. S. Alam, "Bangla language textual image description by hybrid neural network model," *Indonesian J. Electr. Eng. Comput. Sci.*, vol. 21, pp. 757–767, Feb. 2021.

[2] M. A. A. Mumin, A. Awal, M. Shoeb, M. R. Selim, and M. Z. Iqbal, "Sumono: A representative modern Bengali corpus," *SUST J. Sci. Technol.*, vol. 21, no. 1, pp. 78–86, 2014.

[3] N. UzZaman and M. Khan, "A double metaphone encoding for Bangla and its application in spelling checker," in *Proc. Int. Conf. Natural Lang. Process. Knowl. Eng.*, Oct. 2005, pp. 705–710.

[4] R. Sridharamurthy, T. B. Masood, A. Kamakshidasan, and V. Natarajan, "Edit distance between merge trees," *IEEE Trans. Vis. Comput. Graphics*, vol. 26, no. 3, pp. 1518–1531, Mar. 2020.

[5] N. S. Dash and B. B. Chaudhuri, "Corpus based empirical analysis of form, function and frequency of characters used in Bangla," in *Proc. Special Issue Corpus Linguistics Conf.* Lancaster, U.K.: Lancaster Univ. Press, 2001, pp. 144–157.

[6] K. M. Y. A. Majumder, M. Z. Islam, N. UzZaman, and M. Khan, "Analysis of and observations from a Bangla news corpus," in *Proc. 9th Int. Conf. Comput. Inf. Technol., (ICCIT)*, 2006, pp. 520–525.

[7] D. M. F. Khan, "Creation and analysis of a new Bangla text corpus BDNC01," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 5, no. 11, pp. 260–266, Nov. 2017.

[8] M. F. Khan and A. M. Sobhan, "Observations from statistical processing of BDNC01 corpus," *Int. J. Appl. Inf. Syst.*, vol. 3, no. 3, pp. 1–7, 2012.

[9] S. Ahmed, N. Sadeq, S. S. Shubha, M. N. Islam, M. A. Adnan, and M. Z. Islam, "Preparation of Bangla speech corpus from publicly available audio & text," in *Proc. 12th Lang. Resour. Eval. Conf.* Marseille, France: European Language Resources Association, May 2020, pp. 6586–6592. [Online]. Available: https://aclanthology.org/2020.lrec-1.811

[10] D. Goldhahn, T. Eckart, and U. Quasthoff, "Building large monolingual dictionaries at the leipzig corpora collection: From 100 to 200 languages," Natural Lang. Process. Group, Univ. Leipzig, Leipzig, Germany, Tech. Rep., May 2012, pp. 759–765. [Online]. Available: https://aclanthology.org/L12-1154/

[11] M. Biswas, R. Islam, G. K. Shom, M. Shopon, N. Mohammed, S. Momen, and A. Abedin, "BanglaLekha-isolated: A multi-purpose comprehensive dataset of handwritten Bangla isolated characters," *Data Brief*, vol. 12, pp. 103–107, Jun. 2017.

[12] M. T. Alam and M. M. Islam, "BARD: Bangla article classification using a new comprehensive dataset," in *Proc. Int. Conf. Bangla Speech Lang. Process. (ICBSLP)*, Sep. 2018, pp. 1–5.

[13] M. T. Hoque and M. Kaykobad, "Coding system for Bangla spell checker," in *Proc. 5th Int. Conf. Comput. Inf. Technol.*, Dhaka, Bangladesh, 2002, pp. 182–185.

[14] N. Uzzaman and M. Khan, "A Bangla phonetic encoding for better spelling suggestion," in *Proc. 7th Int. Conf. Comput. Inf. Technol.*, Dhaka, Bangladesh, 2004, pp. 1–6.

[15] N. Uzzaman and M. Khan, "A comprehensive Bangla spelling checker," in *Proc. Int. Conf. Comput. Process. Bangla*, Dhaka, Bangladesh, 2006, pp. 1–8.

[16] M. S. Ahmed, T. Goncalves, and H. Sarwar, "Improving Bangla OCR output through correction algorithms," in *Proc. 10th Int. Conf. Softw., Knowl., Inf. Manage. Appl. (SKIMA)*, 2016, pp. 338–343.

[17] P. Mandal and B. M. M. Hossain, "Clustering-based Bangla spell checker," in *Proc. IEEE Int. Conf. Imag., Vis. Pattern Recognit. (icIVPR)*, 2017, pp. 1–6.

[18] S. Sooraj, K. Manjusha, M. A. Kumar, and K. P. Soman, "Deep learning based spell checker for Malayalam language," *J. Intell. Fuzzy Syst.*, vol. 34, no. 3, pp. 1427–1434, Mar. 2018.

[19] T. Mittra, S. Nowrin, L. Islam, and D. C. Roy, "A Bangla spell checking technique to facilitate error correction in text entry environment," in *Proc. 1st Int. Conf. Adv. Sci., Eng. Robot. Technol. (ICASERT)*, May 2019, pp. 1–6.

[20] H. M. M. Hasan, M. A. Islam, M. T. Hasan, M. A. Hasan, S. I. Rumman, and M. N. Shakib, "A spell-checker integrated machine learning based solution for speech to text conversion," in *Proc. 3rd Int. Conf. Smart Syst. Inventive Technol. (ICSSIT)*, Aug. 2020, pp. 1124–1130.

[21] I. Ahamed, M. Jahan, Z. Tasnim, T. Karim, S. M. S. Reza, and D. A. Hossain, "Spell corrector for Bangla language using Norvig's algorithm and jaro-winkler distance," *Bull. Electr. Eng. Informat.*, vol. 10, no. 4, pp. 1997–2005, Aug. 2021.

[22] M. N. Jahan, A. Sarker, S. Tanchangya, and M. A. Yousuf, "Bangla real-word error detection and correction using bidirectional lstm and bigram hybrid model," in *Proc. Int. Conf. Trends Comput. Cognit. Eng.*, M. S. Kaiser, A. Bandyopadhyay, M. Mahmud, and K. Ray, Eds. Singapore: Springer, 2021, pp. 3–13.

[23] M. J. Alam, N. UzZaman, and M. Khan, "N-gram based statistical grammar checker for Bangla and English," in *Center for Research on Bangla Language Processing (CRBLP)*. Dhaka, Bangladesh: BRAC Univ., 2006, pp. 119–122.

[24] K. M. A. Hasan, A. Mahmud, A. Mondal, and A. Saha, "Recognizing Bangla grammar using predictive parser," *Int. J. Comput. Sci. Inf. Technol.*, vol. 3, no. 6, pp. 61–73, Dec. 2011.

[25] M. A. Islam, K. M. A. Hasan, and M. M. Rahman, "Basic HPSG structure for Bangla grammar," in *Proc. 15th Int. Conf. Comput. Inf. Technol. (ICCIT)*, Dec. 2012, pp. 185–189.

[26] R. Rahman, M. Habib, M. Rahman, S. Shuvo, and M. Uddin, "An investigative design based statistical approach for determining Bangla sentence validity," *Int. J. Comput. Sci. Netw. Secur.*, vol. 16, pp. 30–37, Dec. 2016.

[27] R. Z. Rabbi, M. I. R. Shuvo, and K. M. A. Hasan, "Bangla grammar pattern recognition using shift reduce parser," in *Proc. 5th Int. Conf. Informat., Electron. Vis. (ICIEV)*, May 2016, pp. 229–234.

[28] M. M. Rahman, M. Abdulla-Al-Sun, K. M. A. Hasan, and M. I. R. Shuvo, "Designing a Bangla parser using non-deterministic push down automata," in *Proc. Int. Conf. Electr., Comput. Commun. Eng. (ECCE)*, Feb. 2017, pp. 571–576.

[29] S. Islam, M. F. Sarkar, T. Hussain, M. M. Hasan, D. M. Farid, and S. Shatabda, "Bangla sentence correction using deep neural network based sequence to sequence learning," in *Proc. 21st Int. Conf. Comput. Inf. Technol. (ICCIT)*, Dec. 2018, pp. 1–6.

[30] S. F. Shetu, M. Saifuzzaman, M. Parvin, N. N. Moon, R. Yousuf, and S. Sultana, "Identifying the writing style of Bangla language using natural language processing," in *Proc. 11th Int. Conf. Comput., Commun. Netw. Technol. (ICCCNT)*, Jul. 2020, pp. 1–6.

[31] A. S. Prapty, M. R. Anwar, and K. M. A. Hasan, "A rule-based parsing for Bangla grammar pattern detection," in *Proc. Int. Joint Conf. Adv. Comput. Intell.*, M. S. Uddin and J. C. Bansal, Eds. Singapore: Springer, 2021, pp. 319–331.

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, and B. Thirion, "Scikitlearn: Machine learning in Python," *J. Mach. Learn. Research.*, vol. 12, pp. 2825–2830, Nov. 2011.

[33] T. Eckart, U. Quasthoff, and D. Goldhahn, "Language statistics-based quality assurance for large corpora," in *Proc. Asia Pacific Corpus Linguistics Conf.*, Auckland, New Zealand, May 2012.

[34] G. De Marzo, A. Gabrielli, A. Zaccaria, and L. Pietronero, "Dynamical approach to Zipf's law," *Phys. Rev. Res.*, vol. 3, no. 1, Jan. 2021, Art. no. 013084.

[35] I. Bolshakov and D. Filatov, "Distributions of functional and content words differ radically," in *Proc. Mex. Int. Conf. Artif. Intell.*, vol. 4293, Nov. 2006, pp. 838–843.

[36] E. Haryu and S. Kajikawa, "Use of bound morphemes (noun particles) in word segmentation by Japanese-learning infants," *J. Memory Lang.*, vol. 88, pp. 18–27, Jun. 2016.

[37] D. Pinto, D. Vilari no, Y. Alemán, H. Gómez, N. Loya, and H. Jiménez-Salazar, "The soundex phonetic algorithm revisited for SMS text representation," in *Text, Speech Dialogue*, P. Sojka, A. Horák, I. Kopeček, and K. Pala, Eds. Berlin, Germany: Springer, 2012, pp. 47–55.

[38] I. A. Ahmed, "Different types of spelling errors made by Kurdish EFL learners and their potential causes," *Int. J. Kurdish Stud.*, vol. 3, pp. 93–110, Aug. 2017.

[39] Z. Bhatti, I. A. Ismaili, A. A. Shaikh, and W. Javaid, "Spelling error trends and patterns in Sindhi," *J. Emerg. Trends Comput. Inf. Sciences.*, vol. 3, no. 10, pp. 1435–1439, 2012.

[40] R. Dong, Y. Yang, and T. Jiang, "Spelling correction of nonword errors in uyghurchinese machine translation," *Inf.*, vol. 10, no. 6, pp. 202-1–202-9, 2019.

[41] K. M. A. Hasan, M. Hozaifa, and S. Dutta, "Detection of semantic errors from simple Bangla sentences," in *Proc. 17th Int. Conf. Comput. Inf. Technol. (ICCIT)*, Dec. 2014, pp. 296–299.

[42] D. Jurafsky and J. H. Martin, *Speech and Language Processing*. London, U.K.: Pearson, 2008, ch. N-Gram Language Models.

[43] P. Xia, L. Zhang, and F. Li, "Learning similarity with cosine similarity ensemble," *Inf. Sci.*, vol. 307, pp. 39–52, Jun. 2015.

NAHID HOSSAIN received the joint bachelor's degree from Frederick University, Cyprus, and United International University (UIU), Bangladesh, and the master's degree from UIU. He is currently an Assistant Professor with the Computer Science and Engineering (CSE) Department, UIU. Before joining UIU, he worked as a Software Engineer with the Natural Language Processing (NLP) Department, eGeneration Ltd., Bangladesh. His research interests include natural language processing, data mining, big data, and machine learning and AI. He has got several national and international scholarships and awards, including a scholarship from European Union and the Gold Medal from Education Minister of Bangladesh.

SALEKUL ISLAM (Senior Member, IEEE) received the Ph.D. degree from the Computer Science and Software Engineering Department, Concordia University, in June 2008, under the supervision of Dr. J. William Atwood. He is currently a Professor and the Head of the CSE Department, United International University, Bangladesh. Previously, he worked as an FQRNT Postdoctoral Fellow at the Énergie, Matériaux et Télécommunications (EMT) Centre, Institut National de la Recherche Scientifique (INRS), Montréal, Canada. His research interests include future internet architecture, blockchain, edge cloud, software-defined networks, multicast security, security protocol validation, and machine learning and AI. He is serving as an Associate Editor for IEEE Access journal.

MOHAMMAD NURUL HUDA received the Ph.D. degree in automatic speech recognition from Toyohashi University of Technology, Aichi, Japan, in March 2009. He is currently a Professor and the Director of graduate programs with the Computer Science and Engineering Department, United International University, Bangladesh. Moreover, he is the Director of the Natural Language Processing Department, eGeneration Ltd., Bangladesh. His research interests include natural language processing, machine learning, speech recognition, speech synthesis, artificial intelligence, and computational linguistics.