SweetCoat-2D: Two-Dimensional Bangla Spelling Correction and Suggestion Using Levenshtein Distance and String Matching Algorithm.

Md Mahadi Hasan*, David Dew Mallick†,Towhid Khan‡,MD. Mustakin Alam§,Md Humaion Kabir Mehedi¶,Annajiat A
Department of Computer Science and Engineering, School of Data and Sciences (SDS), Brac University, Dhaka, Bangla
**md.mahadi.hasan1@g.bracu.ac.bd,* †*david.dew.mallick@g.bracu.ac.bd,* ‡*towhid.khan@g.bracu.ac.bd,*
§*md.mustakin.alam@g.bracu.ac.bd,* ¶*humaion.kabir.mehedi@g.bracu.ac.bd* ‖*annajiat@gmail.com*

***Abstract*—Autonomous spelling correction and suggestion generation for any language is a tough task since the system must grasp context or it will create a negative impression. Our goal is to reduce this difficulty and provide a more accurate spelling check and suggestion for the Bangla language. Recently, the Bangla language has entered a more advanced phase of its procedure for automatically repairing misspelled words. However, just a few works have been undertaken on this subject that are adequate. In this study, we developed a 2D encoding mechanism dubbed SweetCoat-2D that was utilized in conjunction with Levenshtein Distance and several String matching approaches to evaluate more precisely. Our encoding method outperforms the rest of the process, thus the outcomes are encouraging. We also constructed a corpus of 15,000 misspelled words to which we applied our SweetCoat-2D encoding algorithm in conjunction with the Levenshtein Distance and String Matching algorithms, yielding improved spelling suggestion results of 92.404 percent, where accuracy is calculated only for correct suggestions which are provided at the first index. Therefore, there is a high probability that the presented model will be useful and have a substantial impact on the process of correcting misspellings in Bangla.**

***Index Terms*—Spell, Correction, Encoding, Bangla Spelling, Wrong Words**

I. Introduction

This study aims to develop a spelling correction system for Bangla, a widely spoken language, using a combination of direct dictionary search and the Levenshtein edit distance method with String matching algorithm as well. The system will use a wide vocabulary and a novel method for calculating correctness in order to achieve improved accuracy in spelling correction compared to previous research on this topic. The primary objective of the study is to verify the spelling of Bangla words and correct them with high precision.Communication relies heavily on the written form of a language, whether it is handwritten or typed. When speaking, it is not necessary to know the correct spelling of a word, and communication is possible even without proper grammatical maintenance, but correct spelling is essential when writing. Spelling errors are common, particularly in complex languages such as Bangla. Bangla is the 7th [1] most commonly spoken language worldwide.

Due to the complex nature of the English language, spelling errors are prevalent. Spelling errors are not always the consequence of their complexity; they can also stem from hasty typing or a lack of competence. Due to the prevalence of misspellings, automatic methods for correcting misspellings are crucial. There are numerous works on this subject, yet there are not many notable Bangla works.. Before determining the accuracy, the direct dictionary search and Levenshtein edit distance approach will be utilized to verify the word's accuracy and obtain potential corrections after using String Matching algorithms. There have been prior research employing the edit distance method, but in this study, a far wider vocabulary and a novel method of calculating correctness are applied, providing superior results.

Moreover,sound is formed by the expulsion of air from the lips, it has no shape. Humans speak them, and they are audible to the ear. To aid in the transcription of the language, a number of symbols representing the sounds have been produced. This symbol is known as letter(বর্ণ). This collection of letters constitutes the alphabet.Bengali sounds also come in two varieties: 1. vowels (স্বরবর্ণ) 2. Consonant (ব্যঞ্জনবর্ণ). `Vowels:` Vowels are the written symbols or symbols of vowel sounds. There are seven fundamental vowel sounds in Bengali.However,There are 11 vowels, though.(অ আ ই ঈ উ ঊ ঋ এ ঐ ও ঔ) `Consonants:` Consonants are written signs or symbols representing consonant sounds. The Bengali language features 39 consonants.(ক খ গ ঘ ঙ চ ছ জ ঝ ঞ ট ঠ ড ঢ ণ ত থ দ ধ ন প ফ ব ভ ম য র ল শ ষ স হ ড় ঢ় য় ৎ ং ঃ ঁ) `Abbreviations of Vowels:` When coupled with consonants, nondistinct vowels are shortened to their abbreviated versions. These shortening of vowels are known as "কার." Vowel 'কার'-symbol 10. Namely:(া ি ী ু ,ূ ে ,ৈ ো ৌ)

A. Categories of Error

In the modern era of technology, many errors occur when writing on the gadget. As Bengali is one of the most widely used languages in the world, there are several errors here

as well. According to studies, there are three distinct sorts of errors. In addition to this, there are typically other errors when writing in the Bengali language. Some of these errors are the most prevalent. Our research focuses mostly on these errors.

- Vowel Abbreviation Error This is an error that occurs all too frequently. Due to the similarity in pronunciation, all types of individuals make this error more frequently.ই—কার (ি) in place of ঈ—কার (ী) , In place of উ—কার (ু),ঊ—কার (ূ) and their opposites also happen. Due to these errors, however, the meaning difference is not observed usally.Even then it is one of the huge mistakes in Bengali grammar.For example:

  নাতিশীতোষ্ণ - নাতীশিতোষ্ণ , ভুল - ভূল [2]
  (e.g : vacuum - vacume) [3]

- Consonant Spelling error There is no distinction between appearance and spelling. However, grammarians consider this a grave writing error. For instance: ন in place of ণ, ambiguity of শ, ষ , সas all are sounds like (ssh sound). Similarly, ambiguity of (rr sound) র , ড় , ঢ় and in place of খ , ক্ষ or য ,জ . For example :

  ক্ষমা - খমা , বাড়ি - বারি ,জায়গা - যায়গা [2]
  (e.g : fascinating - facinating)[1]

- Added letter error This is a pretty frequent error. Many educated individuals make these errors unknowingly. Fundamentally, these errors are more prevalent due to the pronunciation of the language.For example:

  রিক্সা - রিকশা , সন্তান - সনতান [4]
  (e.g : publicly – publically)[1]

- Pronunciation mistake Long-term usage of a regional language causes this type of error to occur. People frequently make errors in regional tension when writing. Moreover, since it is simple to say, such errors are more prevalent in slips of the tongue, but many individuals use them in writing.For Example:

  গ্লাস - গেলাস , ক্লাস - কেলাস [4]
  (e.g : tomorrow – tommorow)[1]

II. Literature Review

Numerous published works focus on optimal text encoding and clustering. With that many of them focus on approaches for spell checking, while others focus on classification algorithms. However, comparatively a handful of works are available as a mechanism for assessing Bangla spell checking. Our objective is to create a spelling checker for the Bangla language with our own unique encoding mechanism. The authors investigated the properties of Chinese spelling checks with the Cantonese correction system using the N-gram language model [5]. In addition, this work describes the Knuth-Morris-Pratt (KMP) algorithm. It can identify misspellings and improper usage of Cantonese in a sentence and recommend corrections.They separate their work into two steps, namely the preprocessing phase and the Chinese spelling check using the native corrective phrase. This research study [6] focuses on the construction of an automated transliteration system, which is the act of transforming written text from one language to another, with an emphasis on correct spelling. The technique is designed to be utilized when entering text or when a user is uncertain of the correct spelling of a term. The authors implemented the Levenshtein distance algorithm, which is a measure of the similarity between two strings, and the unigram method, which is a statistical method used to predict the probability of a word occurring in a given language based on the frequencies of its individual characters, in order to improve the accuracy of the spell-checking function. This work [7] provides a clustering-based technique for Bangla spell checking that reduces search space and improves efficiency. The spell checker is designed to handle both typographical and phonetic errors, and its performance is evaluated using 2,450 misspelled words. The results indicate that the proposed method has a 99.8% success rate and outperforms two alternative Bangla spell checkers, Avro and Puspa. Using a combination of algorithms and databases, this research [8] proposes a solution to the problem of spelling errors in the Bangla language. It is suggested that the technology could be utilized efficiently in a general-purpose search engine, making it easier for users to locate the information they require, even if they make spelling mistakes when searching.This research [9] study examines the application of a bidirectional long short-term memory (LSTM) language model for context-sensitive spelling detection and correction in Arabic, a Middle Eastern and North African language.Error detection and correction, which can be computationally costly, are often the two operations that make up spelling correction systems. Neural spelling correction models, which employ artificial neural networks to discover and correct errors, frequently seek to fix errors throughout a whole sentence, which might result in overcorrection. This study, [10] presents a novel method for correcting spelling errors in the left-to-right scripted Bangla language. The framework, which is known as the detector-purifier-corrector technique, employs denoising transformers to overcome existing difficulties in spelling correction. In addition, the researchers devised a way for constructing large-scale corpora from scratch, which assists in overcoming resource constraints for this sort of language. The findings [11] of their empirical research indicate that this strategy is very effective and significantly beats existing state of the art technologies.The proposed technique

[1] https://en.wikipedia.org/wiki/Commonly$_{m}isspelled_{E}nglish_{w}ords$

combines an edit distance algorithm and a probability-based N-gram language model to detect and correct errors at both the word and sentence levels.To evaluate the performance of the proposed method, the authors built a large corpus containing 0.5 million words and 50,000 N-gram sentences and tested it with various test data collected from online sources. The accuracy of the spell checker was found to be approximately 97% according to the authors. This work [12] provides a word-based Bangla spell checker, a tool aimed to improve the quality of suggestions for fixing spelling errors in Bangla, a South Asian language. The authors emphasize that the complexity of the Bangla character set and spelling standards makes it difficult to construct a spell checker for this language. The majority of existing Bangla spell checkers correct errors at the character level, however the authors claim that this technique is insufficient to solve the issues of Bangla spelling errors.

III. Methodology

A. *SweetCoat-2D*

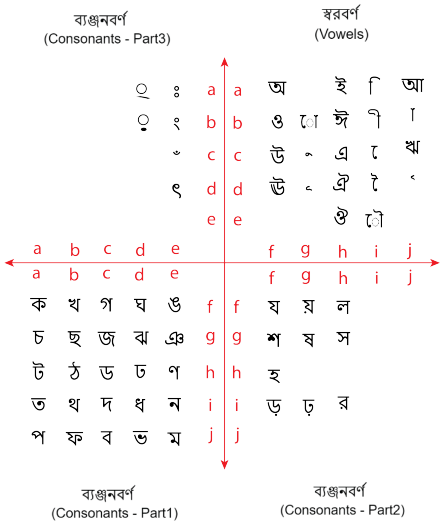

Fig. 1. 2D-Model

Now, we have developed a mechanism to automatically detect such errors. We will primarily cover Bengali with English here. And this job will be completed in 2D. We began by dividing the Bengali alphabet into four sections based on Bengali grammar. The initial section will only contain vowels and their abbreviations. The second section will consist of the most frequent consonant letters. Other consonants in the third section. And in the fourth section, the less frequently used and more dependent on other letters are retained. Consequently, all of our Bengali letters have been put on the 2d axis. Therefore, each letter consists of two English letters.For Example:

This encoding scheme is advantageous. Levenstein edit distance can be precisely determined. For instance, the edit distance between the consonant "ক" and the vowel "ি" is 1. However, nobody makes the error of writing "ি" instead of "ক". Everyone writes "ী" rather than "ি" . Therefore, the recommendation is subpar because the edit distance is equivalent. This model's primary job is to keep the letters of the near misses side by side. In this instance, R's edit distance will be 1 while A's edit distance will be 2. Therefore, it provides very refined and accurate advice.

TABLE I
2D Encoding Result

Sl	Bangla	SweetCoat-2D
1	প	aj
2	খ	bf
3	পাখি	ajbjbfai
4	শালিক	ahebcgdaff
5	আমার সোনার বাংলা	eajjebcj chbbjiebcj hjebjbcgeb
6	পাখি উড়ে যায়	fjebgfda acajdc agebbg

When employing two ascii codes instead of one unicode, there is a further advantage. Although it appears large from the outside. In actuality, the database's size is diminishing.Because,ASCII is a 7-bit character encoding, meaning that each character is represented by 7 bits. This permits 128 unique characters, including letters, numbers, and symbols. Unicode is a character encoding with variable length that can represent the majority of the world's written languages. Depending on the precise character being represented, it uses between 8 and 32 bits to represent each character. This enables a significantly greater range of characters, including tens of thousands of symbols and characters from a variety of scripts.Consequently, ASCII and Unicode differ in the number of bits each character. ASCII employs seven bits each character, but Unicode uses anywhere from eight to thirty-two bits per character.

Algorithm 1 2D Encoding Algorithm

Data:
Result: $ajbjbfai$
begin
  $word \leftarrow$
  $Dictionary \leftarrow$: 2D(x,y), : 2D(x,y) ..... n:2D(x,y)
  $word = list(word)$
  x={}

  for $i \leftarrow 0$ **to** $size(word)$ **do**
    while i *in* $Dictionary$ **do**
      $temp \leftarrow 2D(x, y)$
      $x+ = temp$
    end
  end
  $return\ x$
end

B. *Leveinshtein Distance*

Edit distance is a metric used in natural language processing (NLP) to compare phrases and concepts [13]. It is the least number of single-character insertion, deletions, or substitutions required to turn one string into another. Edit distance is employed often in natural language processing applications such as spell checking, machine translation, and information retrieval. In addition, it could be employed to determine the degree of similarity between two texts or to locate synonyms and related phrases in a massive corpus.Several methods exist for accurately determining edit distance, and research in this field has concentrated on enhancing the speed and accuracy of existing systems as well as developing enlarged variants of the edit distance problem that can accommodate more sophisticated edit procedures. The Levenshtein distance between two strings a, b is shown in equation number (1) and the algorithm is given number 2 .

Algorithm 2 Levenshtein Distance

Data: Strings s and t
Result: Levenshtein distance between s and t
begin
  $m \leftarrow |s|$ $n \leftarrow |t|$ $D \leftarrow m \times n$ matrix **for** $i \leftarrow 0$ **to** m **do**
    $D_{i,0} \leftarrow i$
  end
  for $j \leftarrow 0$ **to** n **do**
    $D_{0,j} \leftarrow j$
  end
  for $i \leftarrow 1$ **to** m **do**
    for $j \leftarrow 1$ **to** n **do**
      if $s_i = t_j$ **then**
        $D_{i,j} \leftarrow D_{i-1,j-1}$
      else
        $D_{i,j} \leftarrow \min(D_{i-1,j} + 1, D_{i,j-1} + 1, D_{i-1,j-1} + 1)$
      end
    end
  end
  return $D_{m,n}$
end

This algorithm 2 calculates the Levenshtein distance between two strings, s and t. The distance is defined as the minimum number of single-character edits (insertions, deletions, or substitutions) required to transform s into t. The algorithm uses a dynamic programming approach to find the optimal solution by filling in a matrix D. The value in each cell of the matrix represents the minimum distance between the substrings of s and t that end at the corresponding positions in the strings.

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

In this equation no. 1 ,The first component of this formula indicates the number of insertion or deletion operations required to convert prefix to an empty string or vice versa.In the second block, the first line denotes deletion and the second line represents insertion. Substitutions are made on the final line. 1 represents the indicator function, which is equal to 0 if a(i) == b(j) and 1 otherwise. By |a|, we represent the length of the string a. lev_(a,b)(i,j) represents the distance between string prefixes – the first I characters of a and the first j characters of b.

C. *Ratcliff/Obershelp String Matching Algorithm*

Ratcliff/Obershelp matching, also known as (Gestalt Algorithm)[2] is a technique for machine learning that use a series of rules to recognize patterns in data.Numerous applications, including image and video identification, natural language processing, and data mining, have utilized Gestalt pattern matching. It has been demonstrated to be effective at identifying patterns in data that are difficult for humans to discern, and it may be used to make predictions about future data based on the identified patterns.This study [14] examines the application of text mining techniques and the Ratcliff/Obershelp algorithm to evaluate the quality of student responses to online exams in the Indonesian education system. Due to the shutdown of educational institutions caused by the Covid-19 pandemic, the study was undertaken in reaction to the move to e-learning and online examinations. The Ratcliff/Obershelp algorithm was used to calculate the similarity value between student answers and a teacher's answer key, and the results of the algorithm were applied to the Examz web application, which was designed to determine the status of answers based on a teacher-specified error tolerance.

$$Similarity = \frac{2K_m}{|s_1| + |s_2|} \tag{2}$$

where :

- k_m = number of matching characters in the string
- | s_1 | and | s_2 | = the lengths of the two strings

This study [15] appears to focus on the use of text mining tools to compare Indonesian vernacular and formal words. The study adds that the use of slang phrases, which are frequently abbreviations or weird acronyms with no

[2] https://en.wikipedia.org/wiki/Gestalt$_{p}attern_{m}atching$

relationship to the words they are meant to represent, can lead to uncertainty in tasks such as sentiment analysis and information retrieval. To address this issue, the study used the Nazief Adriani method, which is an algorithm for changing fundamental forms utilizing Indonesian morphology by removing prefixes and suffixes, and the Ratcliff/Obershelp algorithm, which compares two strings to find their similarity. The study attempts to investigate the similarity between slang words and formal words in order to determine the percentage degree of similarity between the two and whether formal basic words can indirectly represent slang words.

D. *LCS - longest common substring*

Finding the longest substring shared by two or more strings is the longest common substring (LCS) issue. It may be utilized in several disciplines, including biology, linguistics, and computer science. In the domain of computational biology, LCS may be utilized to compare DNA sequences and identify similar regions. LCS techniques have been developed in computer science for matching text and patterns [16]. It has been demonstrated that the time required by these methods is on the order of O(n2) or O(n*m), where n and m are the lengths of the strings being compared.

$$\text{LCS}(i,j) = \begin{cases} 0 & ;\ if\ \ i = 0\ or\ j = 0 \\ LCS\left(X_{i-1}, Y_{j-1}\right)^{x_i} & ;\ if\ i,j > 0\ x_i = y_i \\ MAX\left(LCS\left(X_i, Y_{j-1}\right), LCS\left(X_{i-1}, Y_j\right)\right) & ;\ if\ i,j > 0\ x_i \neq y_i \end{cases} \tag{3}$$

where :

$$Sequence(X_i) = (x_1 x_2 x_3 .....x_m)$$
$$Sequence(Y_j) = (y_1 y_2 y_3 .....y_n)$$

Longest Common Substring (LCS) is the challenge of determining the longest contiguous substring shared by two or more strings. Several strategies have been created to handle this problem, and each has its own tradeoffs between time and space requirements.

Frequently, dynamic programming is employed to overcome the LCS problem. In this approach, a matrix is used to hold the lengths of the longest common substrings of the prefixes of the input texts. The matrix is filled from the bottom up, with the purpose of creating larger and longer common substrings as we progress from the beginning to the conclusion of the strings. All zeros are used to initialize the matrix, and the following formula is then used to fill it with values [16] The purpose of this work [17] appears to be to give a detailed evaluation of various methods for calculating the longest common subsequence (LCS) of two input strings and to investigate their behavior in diverse application contexts. The authors remark that the performance of these methods can vary greatly based on the characteristics of the instance of the problem and the data structures employed in the implementation. In addition, they differentiate between approaches that decide the actual LCS and those that merely calculate its length, as the execution time and space requirements can vary dramatically between tasks. According to the authors, this is the first time such an investigation has been done. They emphasize that the work provides only a high-level summary of the performance of the algorithms, although more in-depth research have been published elsewhere.

E. *Jaro and Jaro Winkler Similarity*

Jaro and Jaro Winkler similarity are extensively used string similarity metrics in information retrieval and natural language processing activities [18]. Jaro similarity is the minimal number of single-character transpositions necessary to turn one string into the other, divided by the length of the strings. Jaro similarity is symmetric, which means that the similarity between two strings is the same regardless of the comparison order. Jaro Winkler similarity is an extension of Jaro similarity that takes the prefixes of the strings under comparison into consideration. It is frequently employed when the strings being compared may be slightly misspelled or otherwise differ significantly, as it gives precedence to the matching characters that come at the beginning of the strings.Similarities between Jaro and Jaro Winkler are frequently utilized in applications like spelling correction, record matching, and text mining. The formula for Jaro similarity is:

$$Jaro(s1, s2) = (\frac{1}{3})(\frac{m}{|s1|} + \frac{m}{|s2|} + \frac{(m-t)}{m}) \tag{4}$$

where:

- s1 and s2 are the two strings being compared
- |s1| and |s2| are the lengths of s1 and s2, respectively
- m is the number of matching characters in the strings
- m is the number of matching characters in the strings
- t is the number of transpositions (i.e. characters that are correctly matched, but in the wrong order)

The formula for Jaro Winkler similarity is:

$$JaroWinkler(s1, s2) = Jaro(s1, s2) + l*p*(1 - Jaro(s1, s2)) \tag{5}$$

where:

- l is the length of the common prefix of s1 and s2 (up to a maximum of 4 characters)
- p is a scaling factor (typically set to 0.1)

IV. Datasets

We utilized this file of over 60 thousand words for the Bangla dictionary dataset. This file is a hybrid combination of two datasets, as mentioned in the link to the original dataset. There are nearly all word types in each letter. Also, this dataset [19] comprising nearly 15,000 incorrect words was utilized for testing reasons. This dataset has the proper words, which is one of its advantages. Similarly, it

contains all bangla letters misspelled words. This has made evaluating the quality of our study much simpler. Here, we discuss the errors in our paper, which are all underlined in this data set. Also, a variety of errors have been identified, which makes the research results comparable and incisive.

V. Implementation

Our model for spelling correction and recommendation consists of multiple steps. First, we identify a word not contained in our database. For this reason, we have created numerous database divisions. One for (অ) and one for (ক), so that each letter has its own partition. Then, we search the database using the axis's initial letter. Because the likelihood of making a mistake with the first letter in Bengali is low. As is the case in English.

If the intended term is not discovered, an error is detected and the vowel forms are separated. Which is known as Lemma in our model's language. for example,

- বাংলাদেশ - ব ং ল দ শ
- পাখি - প খ
- কর্তব্য - ক ত ব

The word is then compared to other entries in the database. In addition, we have not completed their lemma in this sector. This results in an excellent edit distance. The nearest output is then displayed in the result. However, this is not the optimum solution. For the best solution, we must proceed one step farther. In this situation, Gestalt pattern was employed. The string is matched with the 5 outputs closest to it. And following string matching, the highest matching word is displayed as output. Our Proposed model

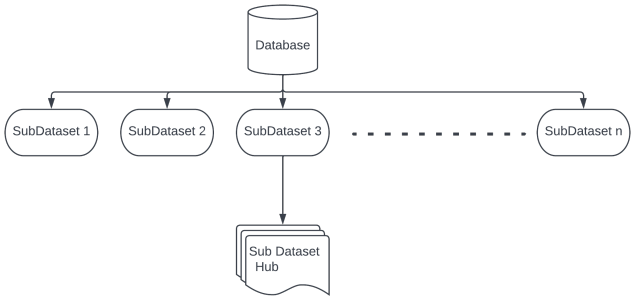

Fig. 2. DataBase Partition

given below:

Finally , We got some outputs like the figure given below.In this figure we can see there are multiple output but index 1 output shows the correct suggestion.

$$Accuracy = \frac{\sum (Correct\ Suggestions\ in\ Index\ (1))}{Total\ Word} \times 100\% \quad (6)$$

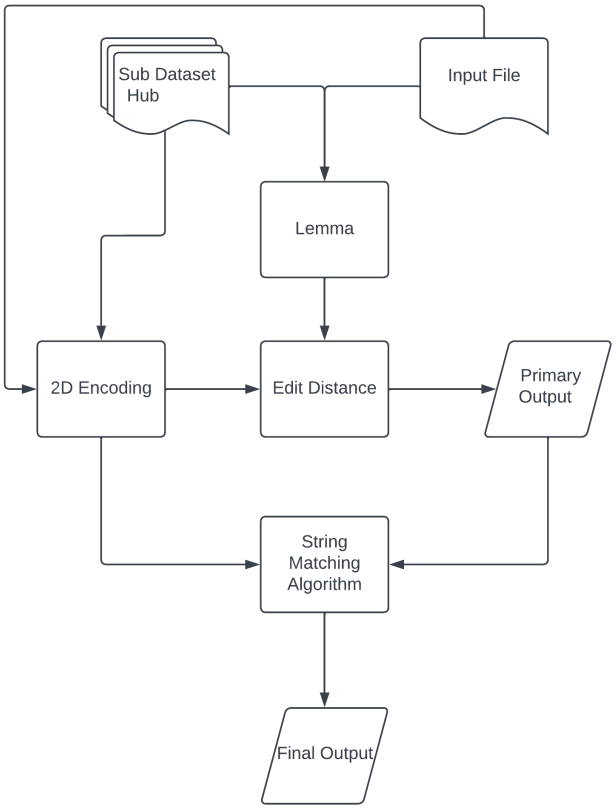

Fig. 3. Our Proposed Model

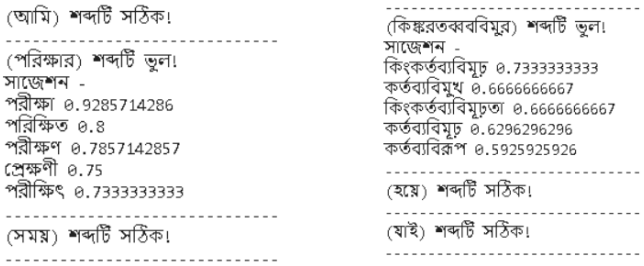

Fig. 4. User Base Output

VI. Result Analysis

Firsly, Wanted to see which algorithm shows best result according to our 2D model . Then we figured out Levenshtein Edit Distance and Gestalt pattern matching outperforms in this case. Then we decided to create Hybrid algorithm with these two algorithm and this time we got better result then before.

The following figure, Fig [9] represents our 2D encoding model performance in comparison with Raw Bangla Encoding and Phonetic Encoding where our encoding outperforms other encoding mechanisms. Initially we steer through 15 thousand wrong words which were constructed by our own. In addition we ran through three different

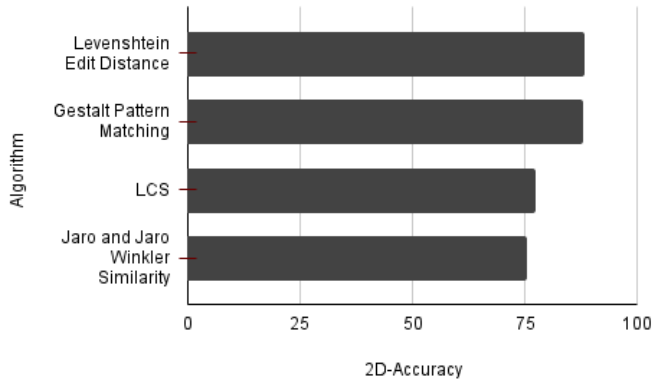

Fig. 5. Comparison With Other Model and String Matching Algorithm

TABLE II
HYBRID ALGORITHM ACCURACY

Algorithms	Accuracy
Levenshtein Edit Distance	88.3129
Gestalt Pattern Matching	87.856
Levenshtein Edit Distance + Gestalt Pattern Matching	92.4042

encoding mechanisms from where we got the following results where Levenshtein edit distance has the highest performance result of 88.312% with our 2D encoding model. Afterwards 84.588% and 82.229% was achieved by Raw Bangla Encoding and Phonetic Encoding respectively.

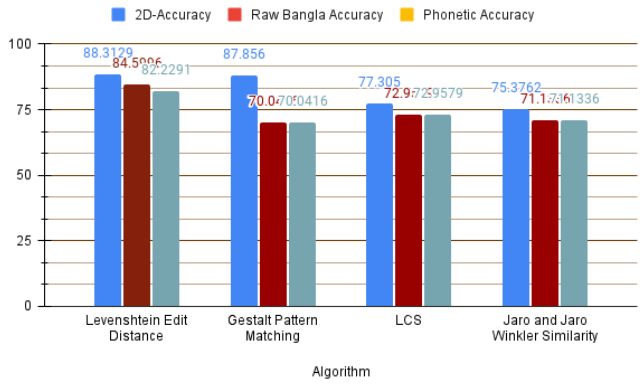

Fig. 6. Comparison With Other Model and String Matching Algorithm

However we took a different pipelining approach with Gestalt Pattern Matching technique where we got 87.856% with our 2D encoding model which was greater than other encoder performance. However 70.041% was achieved by both bangla encoding and phonetic encoding. Then we applied the same procedure with LCS (longest common substring) where our 2D encoding model got a result of 77.305%. Furthermore, Raw bangla encoding and phonetic encoding has the same outcome of 72.957% . Lastly , we conducted an experiment with the Jaro and Jaro Winkler Similarity approach where 75.372% accuracy was achieved with our 2D encoding model . Moreover , Bangla raw encoding and Phonetic encoding also accomplish similar results of 71.1336. Thus we can observe that our 2D encoding model is performing better throughout our experimental procedure.

A. *Comparison with other works*

If we make a comparison table with other works, we can see our model perform better than these works. The comparison table is given below.

TABLE III
COMPARISON WITH OTHER WORKS

Paper Name	Algorithm	Total Word	Accuracy(%)
Our Paper	2D Encoding + Levenshtein Distance + Gestalt Pattern Matching	15,000	92.4042%
[20]	Phonetic Encoding + Edit Distance	1,607	91.67%
[21]	Phonetic Encoding	Not Mentioned	Above 80%
[22]	String Matching	25,000	Showed High Accuracy
[23]	Edit Distance , Stemming	13,000	90.8%
[13]	Edit Distance	150,000	65.17%

VII. CONCLUSION

Therefore, the purpose of this research was to develop a more effective method for correcting Bangla spelling errors involving non-words. The authors proposed a model based on the edit distance method, but with some modifications.

a distinct accuracy calculation method compared to previous papers on Bangla spelling correction. A large dictionary is used, and the model's accuracy is 92.4042%, so it is evident that the model proposed here is promising. This research centered on non-word errors. The authors are currently engaged in another study that will focus on real word errors.

REFERENCES

[1] Lingua, L. o. A. 22, L. o. N. 7, and N. *, ``The 20 most spoken languages in the world in 2022,'' Nov 2022. [Online]. Available: https://lingua.edu/the-20-most-spoken-languages-in-the-world-in-2022/
[2] Bangla Banan Obidhan,, Bangla Academy,Dhaka,Bangladesh.
[3] ``100 most commonly misspelled words.'' [Online]. Available: https://grammar.yourdictionary.com/spelling-and-word-lists/misspelled.html
[4] Bangla Vashar Byakoron, National Curriculum and Textbook Board,Bangladesh.
[5] J.-F. Yeh, L.-T. Chang, C.-Y. Liu, and T.-W. Hsu, ``Chinese spelling check based on n-gram and string matching algorithm,'' in Proceedings of the 4th Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA 2017). Taipei, Taiwan: Asian Federation of Natural Language Processing, Dec. 2017, pp. 35--38. [Online]. Available: https://aclanthology.org/W17-5906
[6] M. M. Hossain, M. F. Labib, A. S. Rifat, A. K. Das, and M. Mukta, ``Auto-correction of english to bengali transliteration system using levenshtein distance,'' in 2019 7th International Conference on Smart Computing & Communications (ICSCC). IEEE, 2019, pp. 1--5.

[7] P. Mandal and B. M. Hossain, ``Clustering-based bangla spell checker,'' in 2017 IEEE International Conference on Imaging, Vision & Pattern Recognition (icIVPR). IEEE, 2017, pp. 1--6.
[8] M. M. Rahman, H. Mahmud, R. S. Rupa, and M. R. Rinty, ``A robust bangla spell checker for search engine,'' in 2021 6th International Conference on Communication and Electronics Systems (ICCES). IEEE, 2021, pp. 1329--1334.
[9] Y. Moslem, R. Haque, and A. Way, ``Arabisc: context-sensitive neural spelling checker,'' 2020.
[10] M. H. Bijoy, N. Hossain, S. Islam, and S. Shatabda, ``Dpcspell: A transformer-based detector-purificator-corrector framework for spelling error correction of bangla and resource scarce indic languages,'' arXiv preprint arXiv:2211.03730, 2022.
[11] T. Mittra, S. Nowrin, L. Islam, and D. C. Roy, ``A bangla spell checking technique to facilitate error correction in text entry environment,'' in 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT). IEEE, 2019, pp. 1--6.
[12] K. Bhowmik, A. Z. Chowdhury, and S. Mondal, ``Development of a word based spell checker for bangla language,'' Ph.D. dissertation, Department of Computer Science and Engineering, Military Institute of ..., 2014.
[13] M. I. K. Islam, R. I. Meem, F. B. A. Kasem, A. Rakshit, and M. T. Habib, ``Bangla spell checking and correction using edit distance,'' in 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT). IEEE, 2019, pp. 1--4.
[14] R. E. Nalawati and A. Dini Yuntari, ``Ratcliff/obershelp algorithm as an automatic assessment on e-learning,'' in 2021 4th International Conference of Computer and Informatics Engineering (IC2IE), 2021, pp. 244--248.
[15] W. Hidayat, E. Utami, and A. D. Hartanto, ``Effect of stemming nazief adriani on the ratcliff/obershelp algorithm in identifying level of similarity between slang and formal words,'' in 2020 3rd International Conference on Information and Communications Technology (ICOIACT), 2020, pp. 22--27.
[16] J. W. Hunt and T. G. Szymanski, ``A fast algorithm for computing longest common subsequences,'' Communications of the ACM, vol. 20, no. 5, pp. 350--353, 1977.
[17] L. Bergroth, H. Hakonen, and T. Raita, ``A survey of longest common subsequence algorithms,'' in Proceedings Seventh International Symposium on String Processing and Information Retrieval. SPIRE 2000, 2000, pp. 39--48.
[18] W. E. Winkler, ``String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage.'' 1990.
[19] M. M. Hasan, ``Misspelling bangla wrong word with answer (15k),'' Jan 2023. [Online]. Available: https://www.kaggle.com/datasets/mahadivai/misspelling-bangla-wrong-word-with-answer-15k
[20] N. UzZaman and M. Khan, ``A comprehensive bangla spelling checker,'' 2006.
[21] ------, ``A bangla phonetic encoding for better spelling suggesions,'' BRAC University, Tech. Rep., 2004.
[22] B. B. Chaudhuri, ``Reversed word dictionary and phonetically similar word grouping based spell-checker to bangla text,'' in Proc. LESAL Workshop, Mumbai, 2001.
[23] M. Islam, M. Uddin, M. Khan et al., ``A light weight stemmer for bengali and its use in spelling checker,'' 2007.