A Context-Sensitive Approach to Find Optimum Language Model for Automatic Bangla Spelling Correction

Muhammad Ifte Khairul Islam[1], Md. Tarek Habib[2],
Md. Sadekur Rahman[3], Md. Riazur Rahman[4]
Department of Computer Science and Engineering
Daffodil International University
Dhaka, Bangladesh

Farruk Ahmed[5]
Department of Computer Science and Engineering,
Independent University, Bangladesh

***Abstract*—Automated spelling correction is an important phenomenon in typing that has intense effect on aiding both literate and semi-literate people while using keyboard or other similar devices. Such automated spelling correction technique also helps students significantly in learning process through applying proper words during word processing. A lot of work has been conducted for English language, but for Bangla, it is still not adequate. All work done so far in Bangla is context-free. Bangla is one of the mostly spoken languages (3.05% of world population) and considered seventh language of all languages in the world. In this paper, we propose a context-sensitive approach for automated spelling correction in Bangla. We make combined use of edit distance and stochastic, i.e. *N*-gram language model. We use six *N*-gram models in total. A novel approach is deployed in order to find the optimum language model in terms of performance. In addition, for finding out better performance, a large Bangla corpus of different word types is used. We have achieved a satisfactory and promising accuracy of 87.58%.**

***Keywords*—Spelling correction; non-word error; N-gram; edit distance; magnifying search; accuracy**

I. Introduction

Spelling error is a common problem in every language whether it is in handwritten or in typing form. Therefore spelling checking and correction is always in the focus of computational linguistics for almost in every language. As a result significant efforts on this area have been observed in various languages like English, Chinese and Arabic. Though Bangla is one of the most widely spoken languages (3.05% of world population) and considered seventh language of all languages in the world [1], not so many notable works were found on automated spelling correction. However, it is also observed that in case of all works of spelling correction in Bangla, context-free spelling checking has been deployed by the researchers. Thus context-sensitive spelling checking remains out of focus in Bangla. Therefore, main focus of this research is to propose a context-sensitive language model. The language model consists of stochastic, i.e. *N*-gram language model and edit distance. Here *N*-gram contributes context-sensitive assessment and edit distance contributes context-free assessment. So, we take advantages of both context-free and context-sensitive approaches in our model. Six *N*-gram based stochastic language models are used. We propose a novel approach for finding the optimum language model. The corpora used so far in the works of automated Bangla spelling correction are not so large. The corpus that we use have surpassed all other so far used Bangla corpus in terms of size.

Rest of the paper is organized as follows: Section II highlights the ongoing researches those were targeted to solve the problems related to spelling mistakes. Next section discusses about the types of spelling errors and also about fundamental ideas of the stochastic language models those are used in this research. Then section IV describes the solution approaches of our research and also about the proposed algorithms. Section V includes the experimentations and results of our findings. Then VI presents the comparisons of our findings with the findings of other researches. Finally section VII concludes the paper mentioning the contribution and limitation of this research.

II. Literature Review

A number of research efforts has been performed in order to solve automated spell checking and correction in different international languages. English, Bangla, Arabic and Chinese are some most spoken languages in the world[1].Notable work on spelling checking in English has been done in [2], [3] and[4]. Bangla is the seventh (7^{th}) most spoken language in the world [1]. Some efforts on automated spell checking in Bangla have been reported in [5], [6], [7] and [8]. Likewise, some efforts on automated spell checking in Chinese [9] and Arabic [10] have been reported. Although almost all of them have concentrated on context-free spell checking, very few of them focused on Context-Dependent spell checking, where a lot of potential of ray of success is lying in.

Automated spell checking in Bangla has been experienced by a small number of papers [5], [6], [7], [8], [11], [12], [13], [14], and [15]. All of them concentrated automated context-free spell checking and correction, whereas none has performed Context-Dependent spell checking and correction in Bangla. Although different techniques are deployed in [5], [6], [7], [8], [11], [12], [13], [14], and [15], one thing is common for them. It is the absence of a balanced, big and reputable corpus. P. Mondal and B.M.M. Hossain [5] have used clustering based on edit distance in order to solve the problem of automated Bangla spell checking. Although they

have claimed to chance an accuracy of 99.8%, their findings are not performed done to the size of test data use, i.e. 2450 words only. They deal with phonetic and typographical errors. N.U. Zaman and M. Khan [8] have used mapping rule based on edit distance and double metaphone in order to deal with automated Bangla spell checking problem. Though they have claimed to have an accuracy 91.67%, their input data is only 1607 words. Bidyut Baran Chaudhuri [11] used string matching algorithm for identifying phonetic errors. At first, he mapped the phonetically similar single unit of character code in a dictionary. He also construct a reversed dictionary which was used to keep characters of each word in reverse order. Misspelled words were corrected using both dictionary. He claimed that his accuracy rate high with 5% false positive detection. But he dealt with mainly phonetical errors. He need double memory space for one dictionary and its' reversed dictionary. N. U. Zaman and M. Khan [7] modified phonetic encoding based on soundex algorithm for matching Bangla phonetic. They also focused only phonetic errors. M. Z. Islam, M. N. Uddin and M. Khan [12] applied stemming algorithm for spell checking. If the stem is not found, then it produces a suggestions list using suggestion generation process. They used edit distance algorithm to find best match. M. K. Munshi et. al. [13] proposed a probabilistic approach for generating the suggestion list of error words using finite state automaton. Authors of [14] used a direct dictionary look up and binary search for detecting error word and generate suggestions using recursive simulation method. Author of [15] used character-based *N*-gram model for checking correctness of a word in Bangla. But they did not correct incorrect words. As none of them focused the context of the sentence while correcting the incorrect word, their accuracy rate can be changed for the Context-Dependent correction in the test sentences. For example, "লোকটা অনাহারে মচা গলে।" here "মচা" is incorrect word and corrected word is "মারা". If we do not consider the context of the sentence their system may be generate words like "পচা", "মশা", "মনা" etc. as suggestion in terms of edit distance and phonetical similarity. Their system may not suggest "মারা" word because it has less phonetically similarity and its' distance with incorrect word is more than other words. But these words are inappropriate with the context of this sentence. In this work the accuracy was calculated in terms of the context of the sentence. The program of this work correct the error word based on the context of the sentence which was never done before in Bangla.

Some papers [2], [3], [4] on English spell checking and correction have been studied. In one of the papers, i.e.in [2] direct dictionary lookup method was used to detect incorrect word and then suggestion list was created using edit distance and frequency of the word. They did not mention their corpus size, accuracy rate and test data size. Andrew Carlson and Ian Fette [3] use *N*-gram and confusion set to correct real-word and non-word error. They use Brown corpus and WSJ corpus and they got 96.1% accuracy for real-word and 96.4% for non-word error. Authors of [4] use tribayes (combination of trigram model and Bayes approach) to correct real-word error. They use brown corpus for train their data and Chinese Learners of English Corpus (ELEC) corpus for test data. They got 86.75% accuracy. Some paper on spell checking for other language have been presented. Author of [9] used edit distance algorithm, soundex algorithm, and combined them with pinyin to check and correct Chinese language spelling. They did not mention their accuracy and test data size. Authors of [10] proposed a system for checking Arabic language spelling using context words and *N*-gram language models. Their corpus size is 41,170,678 words. They used twenty-eight confusion sets for their experiment. Their average accuracy rate is 95.9%. They handle real-word errors and non-word errors both.

III. Types of Spelling Error and Stochastic Language Models

All spelling errors have been classified by Kukich [16] into two types. One is real-word error and the other one is non-word error. Real-word error means the word is not contextually appropriate though it is valid. For example, in the sentence "I eat water.", "eat" is not contextually appropriate but it is a valid word. Similarly, in Bangla, in the sentence "আমি কাক বাড়ী যাব।**",** কাক is not contextually appropriate but it is a valid word. So, a real-world error occurs here. Non-word error means the word is not valid lexically. For example, in the sentence "I wnta to go home", 'wnta' is not a valid word. In the same way in Bangla, 'লারি' is a lexically invalid word in "আমি কাল লারি যাব।". Kukich [16] has offered some more classification of non-word spelling errors. One is cognitive error and the other one is typographical error. Cognitive error occurs when user does not know the spelling of the erroneous word. Typographical error occurs due to typing mistake. For example, **"**আবাসকি**"** is a Bangla word, from which different errors **'**আববাসকি**'**, 'আসকি', 'আমাসকি**'** and **'**আবাকসি**'** are caused by insertion, deletion, substitution and transposition respectively.

To correct the non-word spelling error in a Context-Dependent way, we use stochastic language models, i.e. *N*-gram language models. *N*-gram language model is a type of probabilistic language model where the approximate matching of next item is very high. Probability is based on counting things or word in most cases. The probability of a word depends on the previous word which is called Markov assumption. First-order Markov model called bigram looks immediate previous one word and second-order Markov model is trigram looks immediate previous two words and similarly an *N*-1 Markov model is called *N*-gram language model which looks previous *N*-1 words [17]. Thus, the general equation for this forward *N*-gram approximation to the conditional probability of the next word in a word sequence, $w_1, w_2, \ldots, w_n$, is:

$$P_f(w_n|w_1^{n-1}) \approx P_f(w_n|w_{n-N+1}^{n-1}) \quad (1)$$

If N = 1, 2, 3 in (1), the model becomes forward unigram, bigram and trigram language model, respectively, and so on. If N=1, forward unigram probability is:

$$P_f(w_n|w_1^{n-1}) \approx P_f(w_n|w_n^{n-1}) \quad (2)$$

If N=2, forward bigram probability is:

$$P_f(w_n|w_1^{n-1}) \approx P_f(w_n|w_{n-1}^{n-1}) \quad (3)$$

If N=3, forward trigram probability is:

$$P_f(w_n|w_1^{n-1}) \approx P_f(w_n|w_{n-2}^{n-1}) \quad (4)$$

As like (1), the general equation for backward *N*-gram approximation to the conditional probability of the previous word in a word sequence, w_1, w_2, ….., w_n, ……, w_m is:

$$P_b(w_n|w_{n+1}^{m}) \approx P_b(w_n|w_{n+1}^{n+N-1}) \quad (5)$$

If N = 1, 2, 3 in (4), the model becomes backward unigram, bigram and trigram language model, respectively, and so on.

If N=1, backward unigram probability is:

$$P_b(w_n|w_{n+1}^{m}) \approx P_b(w_n|w_{n+1}^{n}) \quad (6)$$

If N=2, backward bigram probability is:

$$P_b(w_n|w_{n+1}^{m}) \approx P_b(w_n|w_{n+1}^{n+1} \quad (7)$$

If N=3, backward trigram probability is:

$$P_b(w_n|w_{n+1}^{m}) \approx P_b(w_n|w_{n+1}^{n+2}) \quad (8)$$

There is a more other type of *N*-gram based language models that takes the features of forward and backward *N*-gram into account. This a kind of hybrid of forward and backward *N*-gram, which looks immediate *N*-1 words backward and immediate *N*-1words forward. Thus, the general equation for this combined approximation to the conditional probability of the middle word in a word sequence, w_1,w_2, ...,w_n,…,w_m is:

$$P_c(w_n|(w_1^{n-1}w_m^{n+1})) \approx P(w_n|(w_{n-N+1}^{n-1}, w_{n+N-1}^{n+1}) \quad (9)$$

If N = 1, 2, 3 in (8), the model becomes combined unigram, bigram and trigram language model, respectively, and so on.

If N = 1, combined unigram probability is:

$$P_c(w_n|(w_1^{n-1}w_m^{n+1})) \approx P(w_n|(w_n^{n-1}, w_n^{n+1}) \quad (10)$$

If N = 2, combined bigram probability is:

$$P_c(w_n|(w_1^{n-1}w_m^{n+1})) \approx P(w_n|(w_{n-1}^{n-1}, w_{n+1}^{n+1}) \quad (11)$$

If N = 3, combined trigram probability is:

$$P_c(w_n|(w_1^{n-1}w_m^{n+1})) \approx P(w_n|(w_{n-2}^{n-1}, w_{n+2}^{n+1}) \quad (12)$$

IV. Research Methodology

Our proposed approach handles all kinds of non-word errors. Direct dictionary lookup method is used to detect a non-word error. To correct the misspelled word, minimum edit distance method and *N*-gram language model are combinedly used. Six *N*-gram language models, forward bigram, forward trigram, combined bigram, combined trigram, backward bigram, backward trigram, are used separately. After detecting a misspelled word, *N*-gram probability and minimum edit distance for a candidate correction are calculated. *N*-gram probability will contribute for context in further calculations, on the other hand to estimate structural similarity between misspelled word and candidate corrections minimum edit distance is used. It measures the minimum number of total operations required to transform one string into the other. The operations can be insertion, deletion and/or substitution. To calculate edit distance, we use the minimum edit distance dynamic programming algorithm [17] as written in Algorithm 1.Algorithm 1 works by creating a distance matrix with one column for each symbol in the predicted word sequence and one row for each symbol in the error word sequence in order to compare sequence. By using dynamic programming, Algorithm 1 calculates the minimum edit distance, i.e. Levenshtein distance [17], where it is assumed that insertion and deletion each has a cost of 1 and substitution has a cost of 2.

Algorithm. 1. Algorithm for calculating minimum edit distance

```
min_distance(misspelled_word ,
candidate_correction )

m ← length(misspelled_word)
n ← length(candidate_correction)
create distance matrix dis[n+1,m+1]
for each column i ← 0 to n
for each row j ← 0 to m
    dis[i,j]←min(dis[i-1,j]+ins-
        cost(candidate_correction_j),
        distance[i-1,j-1]+
        subst-cost(misspelled_word,
        candidate_correction_i),
        distance[i,j-1)+
        ins-cost(misspelled_word_j))
```

After finding the minimum edit distance $d(\tilde{w}, w_n)$ between the misspelled word $\tilde{w}$ and a candidate correction w_n, we normalize the distance using (13).

$$\bar{d}(\tilde{w}, w_n) = \frac{d_{max}+1-d(\tilde{w},w_n)}{d_{max}} \quad (13)$$

After normalization, the value of distance ($\bar{d}$) ranges in [1/d_{max}, 1]. If the distance d is maximum then the value of normalized distance $\bar{d}$ is 1/d_{max} and if the distance d is minimum then the value of normalized distance $\bar{d}$ is 1. In our work, maximum distance is 9 and minimum distance is 1. Thus, *N*-gram probability and minimum edit distance of candidate corrections are calculated, where *N*-gram probability takes context into account and minimum edit distance works context-independently. So, the final score $S_c(w_n)$ of a candidate correction w_n considers both the effects of context dependence, i.e. *N*-gram probability $P(w_n)$ and context independence, i.e. minimum edit distance $D(w_n)$ in the way shown in (14).

$$S_c(w) = (1-\alpha)P(w_n) + \alpha D(w_n), where\ 0 \leq \alpha \leq 1 \quad (14)$$

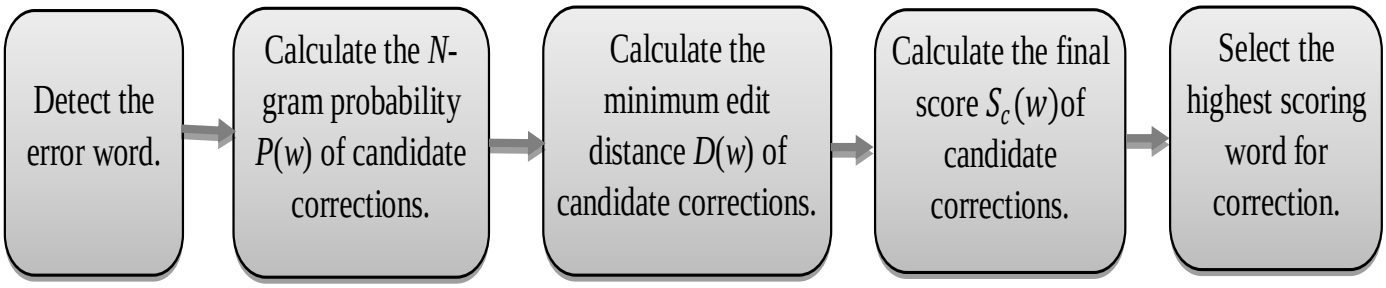

Fig. 1. The Approach for Detecting and Correcting Misspelled Word.

After scoring all candidate corrections the system predicts the word with the highest score as the correct. Suppose the predicted word is $\breve{w}$, then the equation for this word can be written as

$$\breve{w} = \text{argmax}_w S_c(w) \quad (15)$$

The entire process of detection and correction of error word is shown in Fig. 1.

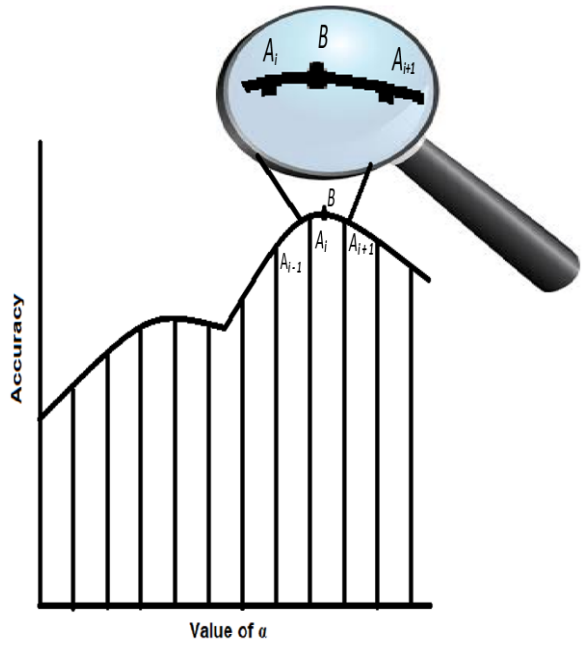

Fig. 2. Underlying Justification Ground for Magnifying Search.

It is easily palpable from (14) that the value of $S_c(w)$ is between 0.0 to 1.0 inclusive since the value of α ranges between 0.0 to 1.0 inclusive. The issue arises from (14) is that what the optimum value (α^*) of α is; that means what the value of α is for which the maximum accuracy is obtained. For this reason, we develop Algorithm 2. We named this Algorithm 2 magnifying search Algorithm. Let us discuss the justification of naming as well as working principle of this algorithm. Suppose that accuracy obtained is represented by A. Hence $A = f(\alpha)$. If we plot these two quantities A andα along x-axis and y-axis, we will obviously get a curve, namely accuracy curve, which will have one or more maximum points. For example, we get an accuracy curve as shown in Fig. 1. Now, we measure the A-values of some equally distant points in order to find the tentative maximum. Of course, we use very small distance; the final maximum will be the tentative one or a point left or right to this tentative point. This is the place where magnifying process comes into play. We magnify the curve fragments left and right to the tentative maximum in order to find more accurate value. We repeat this process until sufficient progress is not made. We can apply the same concept if more than tentative points are found. These entire scenarios are shown in Fig. 2, where A_i is the tentative maximum, $A_{i-1}A_i$ and A_iA_{i+1} are the two curve fragments to the left and right of A_i, respectively. The entire curve fragment $A_{i-1}A_{i+1}$is magnified here.

Algorithm. 2. Magnifying search algorithm for finding α^*, the optimal value of α for each LM.

```
α* ← 0.0
acc_pre ← Accuracy of LM using α = α*
acc_max ← acc_pre
for i ← 0.01 to 1.0 increasing by .01
  acc_cur ← Accuracy of LM for α = i
      if acc_cur > acc_max
            acc_max ← acc_cur
            α*=i
      else if acc_cur = acc_max
              List.add(i)
End for loop

for each element x is in List
      t = Magnify(x, acc_max, 0.001)
      acc_t ← Accuracy of LM for α=t
      if acc_t> acc_max
            acc_max= acc_t
            α*=t
End for loop
```

After calculating $\alpha^*$and accuracy for each of the six language models, i.e. forward bigram, forward trigram, backward bigram, backward trigram, combined bigram and combined trigram, the language model with the highest accuracy is considered as the optimum language model.

Algorithm. 3. Magnify algorithm for magnifying search area for accurate value

```
Magnify(α*, acc_max, δ)
 acc_initial ← acc_max
  for i ← α*+δ, j ← α*-δ; i< α*+(δ*10),
       j<α*-(δ*10); i=i+δ, j=j-δ
  acc_i ← Accuracy of LM for α=i
  if acc_i > acc_max
        acc_max ← acc_i
        α* =i
  acc_j ← Accuracy of LM for α=j
if acc_j > acc_max
        acc_max ← acc_j
    α* = j
 End for loop
 Δ ← |acc_initial − acc_max|
 if Δ ≤ ε
    α* = α*
    return α*
 else
  δ ← δ/10
return Magnify(α*, acc_max, δ)
```

V. Experimentation

A set of training modules were developed to train the six candidate language models, namely forward bigram, forward trigram, combined bigram, combined trigram, backward bigram and backward trigram. All these models are trained based on a corpus. In our work, we have used a very large Bangla corpus, which was constructed from the popular Bangla newspaper the "Daily Prothom Alo." The corpus contains more than 11 million (11,203,790) words and about 1 million (937,349) sentences, where total number of unique word s is 294,371, average *w* word length ($|\overline{w}|$) is 7 and average sentence length is 12. During training, the entire corpus is divided into two parts, namely training part and testing part. The holdout method [18] is used to split the corpus at the proportion of two-third for training and one third for testing. Therefore, this work starts with a training corpus of size more than six (6) hundred thousand sentences. In order to avoid model over-fitting problem (i.e. to have lower training error but higher generalization error), a validation dataset is used. In accordance with this approach, the original training data is divided into two smaller subsets. One of the subsets is used for training, while the other one (i.e. the validation set) is used for calculating the generalization error. Two thirds of the training set are fixed for model building while the remaining one-third is used for error estimation. The test data size is more than 3 million (3,734,596) words and about 300 thousand (312,449) sentences where in every sentence there is a misspelled word in different position in the sentences. The holdout method is repeated for five times in order to find the best model for each candidate models. After finding out the best model, the accuracy of the model is computed using the test set, through which the optimum value (α*) is determined based on magnifying search algorithm as given in Algorithm 2 and Algorithm 3.Table 1 shows the values of α* and accuracy for the best model for each candidate models. The accuracy comparison of all the models is presented in Fig. 3, where the optimum value of α of each model is marked with α*.

From Table 1 and Fig. 3, it can be observed that forward bigram language model generates highest accuracy rate 87.578%, where the value of α^* is .75. So, we can claim the forward bigram to be the optimum model. Other models have also shown good accuracy except combined trigram, which gives an accuracy of only 39.68%. Backward bigram shows an accuracy of 85.964%, which is near to the highest obtained accuracy 87.59%. It is also observed from the Table 1 and Fig.3 that as the value of α increase, the accuracy also increases for all models except for backward trigram. The value of α starts increasing just before α equals α^* for backward trigram. After reaching α*, the accuracy starts decreasing slightly and then remains same for all models other than for backward trigram, for which accuracy starts decreasing.

TABLE I. Optimum value of a (α*) and accuracy rate of all language models

Language Model	**Optimal value of α(α*)**	**Accuracy (%)**
Forward Bigram	0.750	87.59
Forward Trigram	0.656	78.64
Combined bigram	0.658	72.75
Combined trigram	0.90	39.68
Backward Bigram	0.750	85.96
Backward Trigram	0.505	60.54

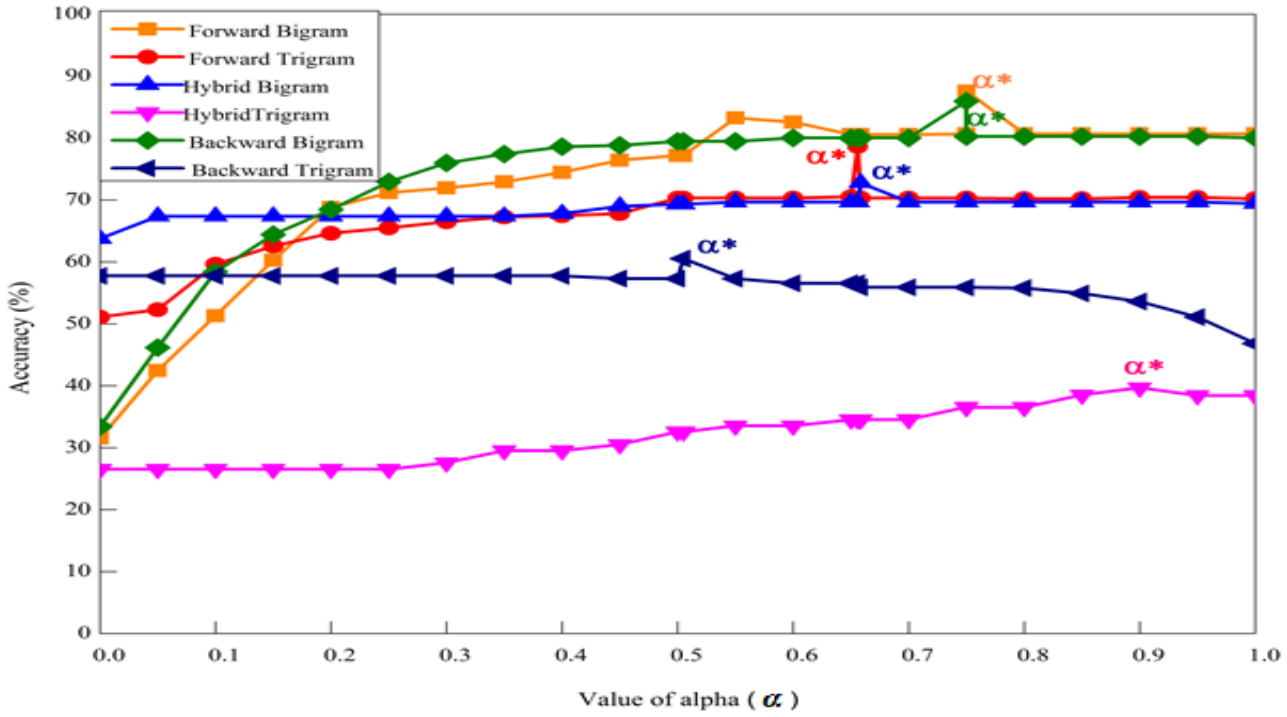

Fig. 3. All Models' Accuracy Comparison Against the Value of α.

TABLE II. ALL MODELS' ACCURACY ACROSS THE MISSPELLED WORD POSITION IN SENTENCES

Misspelled word position	Forward Bigram	Forward Trigram	Combined bigram	Combined trigram	Backward Bigram	Backward Trigram
1	20.24%	30.49%	20%	31.46%	57.28%	70.68%
2	55.36%	52.56%	50.95%	20.0%	53.21%	83.95%
3	55.56%	54.72%	64.03%	21.44%	78.70%	70.64%
4	80.54%	63.57%	59.83%	39.87%	82.36%	76.91%
5	85.634%	75.51%	44.06%	42.13%	87.35%	77.54%
6	68.53%	83.71%	81.96%	48.18%	86.33%	65.37%
7	67.46%	68.23%	86.84%	35.66%	77.70%	64.18%
8	71.85%	65.58%	79.23%	45.29%	79.99%	58.19%
9	73.77%	56.23%	77.05%	25.74%	69.43%	53.53%
10	87.52%	47.07%	54.56%	29.10%	91.72%	47.56%
11	94.66%	85.71%	85.71%	36.29%	45.65%	41.29%
12	95.33%	83.19%	28.51%	28.51%	25.79%	27.66%
Total	87.58%	78.64%	72.75%	39.68%	85.96%	60.54%

In addition, a detailed investigation is conducted, as shown in Table 2, in order to assess the rigorousness of performances of each best candidate language model by varying the misspelled words position in test sentences. The comparison of the six language model's accuracy against the misspelled words position in the test sentences is shown in Fig. 4. From the Fig. 4, it is seen that if misspelled word position is towards the beginning of the sentence then backward bigram, backward trigram and combined trigram show good accuracy rate, but if word position is towards the end forward bigram, forward trigram and combined bigram show better accuracy rate. For middle positions of the sentence all model show good accuracy rate. Combined bigram language model shows almost same accuracy for all positions in the sentence. It can be easily comprehend that if we average the accuracy for all positions, misspelled word forward bigram gives highest accuracy rate of 87.58%.

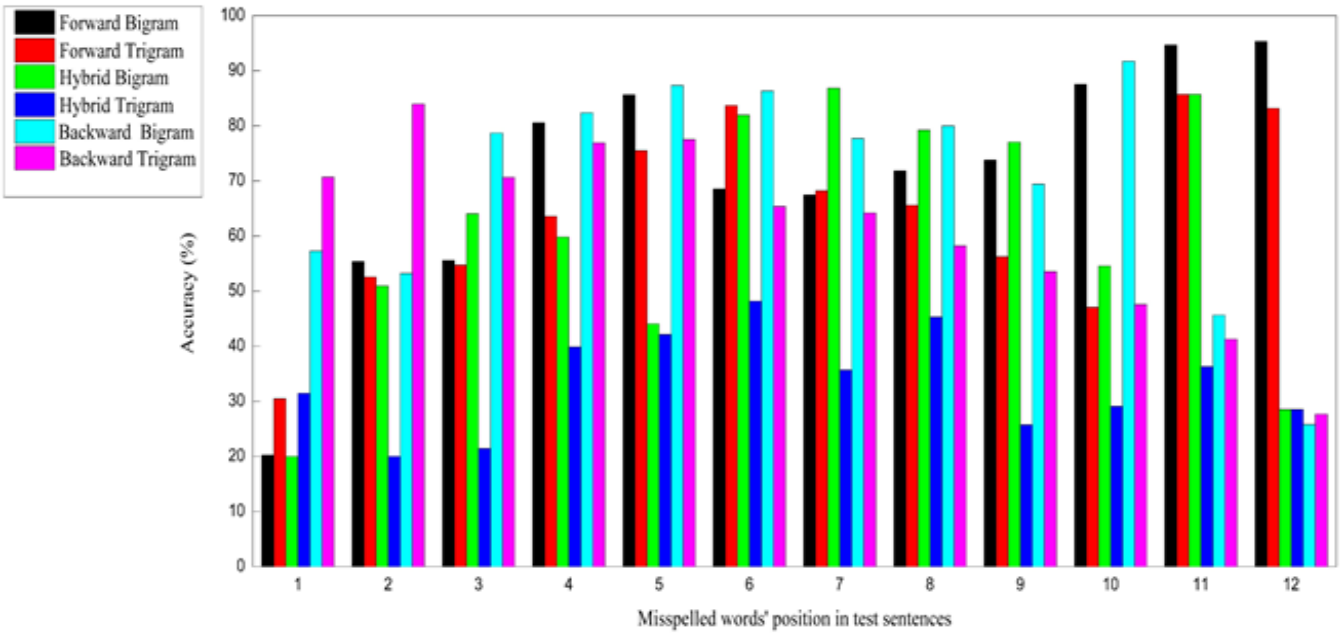

Fig. 4. All Models' Accuracy Comparison Against the Misspelled Words Position.

TABLE III. THE COMPARATIVE NITTY-GRITTY DETAILS OF ALL WORKS REPORTED

Work/Article	Algorithm	Test Data Size	Accuracy	Type of Errors Handled
This work	Context-sensitive technique based on *N*-gram and edit distance	3,734,596 words and 312,449 sentences	87.58%	Non word error
[5]	Clustering	2450	99.8%	Phonetic, typographical
[6]	2-edit distance and phonetic encoding	1607	91.67%	phonetic, typographical, OCR generated
[7]	Phonetic encoding	*NM*	More than 80%	Phonetic, Typographical
[8]	1.Double metaphone 2. Mapping rule.	1607	Edit Distance 0: Correct:1473 Error:134 Accuracy:91.67% Error: 8.33% Error: 107 for edit distance1 Error: 27 for edit distance 2	orthographic rules in Bangla
[11]	String matching algorithm	25,000 words	high accuracy with 5% false positive detection	Phonetic error
[12]	Stemming algorithm and Edit distance	13,000 words	90.8% for correcting single error misspellings and 67% for correcting multiple error misspellings	complex orthographic rules
[13]	Finite state automaton	291 words	92% for correcting single character misspellings and 70% for correcting multiple character misspellings	substitution errors, insertion errors
[14]	Direct dictionary look up method and Recursive Simulation algorithm	*NM*	*NM*	typographical errors and cognitive phonetic errors
[15]	*N*-gram Model (character based)	50,000 correct words and 50,000 incorrect words	96.17%	Non word error

**NM Not mentioned*

VI. COMPARATIVE ANALYSIS OF RESULTS

It is a matter of fact that automated spell checking in Bangla has been performed in a small number of works. Moreover, all of them concentrated automated context-free spell checking and correction, but none of them has performed context-dependent spell checking and correction in Bangla. The size of test data they used is not so big. Some of them achieved good results, whereas some achieved results, which are not up to the mark. Although different context-free

techniques are deployed by them, one thing is common for them. It is the absence of a balanced, big and reputable corpus. In such situation, it is difficult to compare performances obtained by all. In this circumstance, it will not be a callow statement that achieving an accuracy of 87.58% by applying a context-sensitive technique with a training and test data set of big size is quite satisfactory as well as promising. Table 3 shows the comparative nitty-gritty details of all works reported.

VII. Conclusion and Future work

The aim of the research was to find the optimum language model that can assist to overcome Bangla spelling error based on the context. For the purpose of the research a rich and large Bangla Corpus has been used and by applying machine learning techniques on that corpus six language models have been trained for finding the optimum language model for automatic Bangla spelling correction. Finding this language model is the main contribution of the research. Moreover, the approach used for finding the optimum solution is quite novel. Another notable feature of the research is using a large data set for training and testing the model. The accuracy of the model is 87.58% which is good as well as promising. There remains future work for offering a set of corrections rather than offering a single word. Work is in progress to come up with this feature.

References

[1] List of languages by number of native speakers, Available at: https://en.wikipedia.org/wiki/List_of_languages_by_number_of_native_speakers. (Last Accessed: March 10, 2018).

[2] Shashi Pal Singh, Ajai Kumar, Lenali Singh, Mahesh Bhargava, KritikaGoyal, Bhanu Sharma, “Frequency based Spell Checking and Rule based Grammar Checking,” in International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT) 2016.IEEE,2017, pp 4435 – 4439.

[3] Andrew Carlson,Ian Fette, “Memory-Based Context-Dependent Spelling Correction at Web Scale,” in Sixth International Conference on Machine Learning and Applications (ICMLA 2007), IEEE 2007.

[4] Ya Zhou, Shenghao Jing, Guimin Huang, Shaozhong Liu, Yan Zhang ,“ A Correcting Model Based on Tribayes for Real-word Errors in English Essays,” 2012 Fifth International Symposium on Computational Intelligence and Design.

[5] Prianka Mandal, B M Mainul Hossain, “Clustering-based Bangla Spell Checker,” in 2017 IEEE International Conference on Imaging, Vision & Pattern Recognition (icIVPR). IEEE 2017, pp 1-5.

[6] N. UzZaman and M. Khan, “A comprehensive bangla spelling checker,” In the Proceeding of the International Conference on Computer Processing on Bengali (ICCPB), Dhaka, Bangladesh, 2006.

[7] N. UzZaman and M. Khan, “A bangla phonetic encoding for better spelling suggesions,” in Proc. 7th International Conference on Computer and Information Technology, Dhaka, 2004.

[8] N. UzZaman and M. Khan, “A double metaphone encoding for bangla and its application in spelling checker,” in 2005 International Conference on Natural Language Processing and Knowledge Engineering. IEEE, 2005, pp. 705–710.

[9] DongHui Li, DeWeiPeng. “Spelling Correction for Chinese LanguageBased on Pinyin- Soundex Algorithm,” Internet Technology and Applications (iTAP), 2011.

[10] Majed M. Al-Jefri, Sabri A. Mahmoud. “Context-Dependent Arabic Spell Checker using Context Words and N-gram Language Models,” 2013 Taibah University International Conference on Advances in Information Technology for the Holy Quran and Its Sciences, 2013.

[11] Chaudhuri, BidyutBaran, “Reversed Word Dictionary and Phonetically Similar Word Grouping based Spell-checker to Bangla Text,” Proc. LESAL Workshop, Mumbai. 2001.

[12] Md. Zahurul Islam, Md. Nizam Uddin and Mumit Khan, “A Light Weight Stemmer for Bengali and its Use in Spelling Checker,” Proc. 1st Intl. Conf. on Digital Comm and Computer Applications (DCCA07), Irbid, Jordan, March 19-23, 2007.

[13] M. K. Munshi, M. Z. Islam, and M. Khan. "Error-tolerant Finite-state Recognizer and String Pattern Similarity Based Spelling-Checker for Bangla," Proceeding of 5th International Conference on Natural Language Processing (ICON). 2007.

[14] Abdullah, A. B. A., and Ashfaq Rahman. "A Generic Spell Checker Engine for South Asian Languages." Conference on Software Engineering and Applications (SEA 2003). 2003.

[15] Khan, Nur Hossain, et al. "Checking the Correctness of Bangla Words using N -Gram." International Journal of Computer Application 89.11 (2014).

[16] K. Kukich, “Techniques for automatically correcting words in text,” ACM Computing Surveys (CSUR), vol. 24, no. 4, pp. 377– 439, 1992.

[17] Daniel Jurafsky and James H. Martin, (2000), “Speech and Language processing,” USA: Prentice-Hall, Inc.

[18] P.-N. Tan, M. Steinbach, and V. Kumar, “Introduction to Data Mining,” Addison-Wesley, 2006.