International Journal of Electrical and Computer Engineering (IJECE)
Vol. 10, No. 3, June 2020, pp. 3244~3252
ISSN: 2088-8708, DOI: 10.11591/ijece.v10i3.pp3244-3252

An exploratory research on grammar checking of Bangla sentences using statistical language models

Md. Riazur Rahman, Md. Tarek Habib, Md. Sadekur Rahman,
Gazi Zahirul Islam, Md. Abbas Ali Khan
Department of Computer Science and Engineering, Daffodil International University, Bangladesh

Article Info

Article history:

Received Jun 30, 2019
Revised Nov 4, 2019
Accepted Dec 7, 2019

Keywords:

Grammar checking
Language models
Natural language processing
N-grams
Smoothing

ABSTRACT

N-gram based language models are very popular and extensively used statistical methods for solving various natural language processing problems including grammar checking. Smoothing is one of the most effective techniques used in building a language model to deal with data sparsity problem. Kneser-Ney is one of the most prominently used and successful smoothing technique for language modelling. In our previous work, we presented a Witten-Bell smoothing based language modelling technique for checking grammatical correctness of Bangla sentences which showed promising results outperforming previous methods. In this work, we proposed an improved method using Kneser-Ney smoothing based n-gram language model for grammar checking and performed a comparative performance analysis between Kneser-Ney and Witten-Bell smoothing techniques for the same purpose. We also provided an improved technique for calculating the optimum threshold which further enhanced the the results. Our experimental results show that, Kneser-Ney outperforms Witten-Bell as a smoothing technique when used with n-gram LMs for checking grammatical correctness of Bangla sentences.

Corresponding Author:

Md. Riazur Rahman,
Department of Computer Science and Engineering,
Daffodil International University,
4/2, Sobhanbag, Mirpur Road, Dhanmondi, Dhaka-1207, Bangladesh.
Email: riazur_rahman@daffodilvarsity.edu.bd

1. INTRODUCTION

The field of study that deals with processing natural languages is called Natural Language Processing (NLP) which investigates how computers can be used to recognize and operate natural languages [1]. NLP is an important brach of Artificial Intelligence (AI), which has plenty of applications as other branches of AI do like rice grain classification [2], anomalous sound event detection [3], robotic navigation [4], recommendation system for buying house [5], and so on. One such application of NLP is grammar checking [6]. Though there are a lot of tools and techniques, as described in [7-10], developed for grammar checking in recent years, but, grammar checkers have quite a lot of limitations still now.

There are mainly two approaches to implement a grammar checker, namely rule-based approach [11] and statistical approach [12]. In rule-based grammar checkers, a set of manually developed grammatical rules are used to decide the correctness of the given text and developing such rules requires time and high-level linguistic expertise of the target language. Whereas, in statistics-based approach, the grammar rules are built from a text corpus of the target language using statistical methods where common sequences that occur often can be considered correct and the uncommon ones incorrect. Language model (LM) is a widely used statistical technique that builds a statistical machine from a text corpus of the target language that can estimate the distribution of the language as accurately as possible. A central issue in LM estimation

is data sparseness, in which case LMs fails to approximate accurate probabilities due to limited training data. Smoothing [13] is a technique that resolves this problem by adjusting the maximum likelihood estimator to compensate for data sparseness. In practice, LMs are usually implemented in conjunction with smoothing techniques for better performance. There are many smoothing techniques available out of which Witten-Bell (WB) [14] and Kneser-Ney (KN) [15] are by far the two most effective and widely used smoothing techniques.

A number of good works is done in Bangla in different problem domains of NLP, e.g. autocomplete [16], autocorrection of spelling [17], word prediction [18]. Furthermore, there has been much development in grammar checking research in many different languages. Nevertheless, being one of the top ten spoken languages in the world [19], there has been little development in the Bangla language processing specially in grammar checking. Though some efforts have been made, there are still plenty of rooms for improvement. In [20] the authors presented an n-gram LM to design a Bangla grammar checker, where the n-gram probability distributions of parts-of-speech (POS) tags of words are used as feature. A sentence is detected as grammatically correct if the product of all the n-grams in the sentence is greater than zero otherwise incorrect. Due to this, their method suffers from the data sparsity problem, which severely degrades the performance of the system. Moreover, they used a very small corpus of only 5000 words to build the n-gram model and tested the model on a test set of simple sentences. The authors in [21] presented another n-gram based statistical technique for grammar checking. Rather than using probability of POS tags of words this time n-gram probability distribution of words is used to train and test the system. To deal with sparsity problem of n-gram models, they used WB smoothing with their n-gram model. They trained their statistical n-gram model with a small experimental corpus of 1 million words with a test set of 1000 correct and 1000 incorrect sentences. However, their approach did not clarify how the threshold between correct and incorrect sentences is determined which is not a practical approach. Moreover, in our previous work [22], a statistical method was proposed which used n-gram based LM combined with WB smoothing and backoff technique to determine the grammatical correctness of simple Bangla sentences, which presented promising results. Nevertheless, there are still room for improvement and further analysis are required to find an enhanced, robust and well performing statistical grammar checking system for Bangla.

The issues mentioned above and facts motivated this work where a comprehensive comparative study on the performance of WB and KN smoothing based LMs for the purpose of grammar checking of Bangla sentences has been performed to find the best possible LM, settings and methods for the development of a more accurate and robust grammar checker for Bangla. The presented technique was trained on a large Bangla corpus of 20 million words collected from various online newspapers. An improved strategy is proposed to determine appropriate threshold to distinguish between grammatical and ungrammatical sentences. The threshold was finalized by performing cross validation on the training set and testing on a separate validation set in two stages to ensure maximum optimality. The proposed method was tested on an updated realistic and challenging test set of 15000 correct and 15000 incorrect sentences consisting of all kinds of simple & complex sentences with varying lengths. The rest of the paper is organized as follows; section 2 presents some theoretical background on n-gram based sentence probability calculation. Whereas section 3 describes the methodology used for developing the system. Section 4 presents the experimental results while section 5 concludes the paper.

2. STATISTICAL LANGUAGE MODELING

N-gram statistical LMs are very popularly used statistical methods for solving various NLP problems.

2.1. N-gram language models

A language model (LM) is a probability distribution over all possible sentences or strings in a language. Let's assume that S denotes a sentence consisting of a specified sequence of words such that $S = w_1\ w_2\ w_3 \ldots\ w_k$. An n-gram LM considers the word sequence or sentence to be a Markov process [23]. Its probability is calculated as,

$$P(S) = \prod_{i=1}^{k} P(w_i | w_{i-n+1} \ldots w_{i-1}) \quad (1)$$

where n refers to the order of the Markov process. When n = 3 we call it a trigram LM which is estimated using information about the co-occurrence of 3-tuples of words. The probability of $P(w_i | w_{i-n+1} \ldots w_{i-1})$ can be calculated as,

$$P(w_i | w_{i-n+1} \ldots w_{i-1}) = \frac{C(w_{i-n+1} \ldots w_i)}{\sum_w C(w_{i-n+1} \ldots w_{i-1} w_i)} \quad (2)$$

where, $C(w_{i-n+1} \dots w_i)$ is the count of occurrences of word sequence $w_{i-n+1} \dots w_i$ and $\sum_w C(w_{i-n+1} \dots w_{i-1} w_i)$ indicates the sum of counts of all the n-grams that starts with $w_{i-n+1} \dots w_{i-1}$.

For example, let us consider the following Bangla sentence,

কাদের একটি আম খেয়েছে [english] Kader ate a mango
(Kader ekti aam kheychey)

The probability of this sentence can be calculated using bigram LM with (1) as,

*P(কাদের একটি আম খেয়েছে) = P(কাদের|<s>) * P(একটি|কাদের) * P(আম|একটি) * P(খেয়েছে|আম) * P(</s>| খেয়েছে)*

For the same English sentence,
*P(Kader ate a mango) = P(Kader |<s>) * P(ate | Kader) * P(a| ate) * P(mango |a) * P(</s>| mango)*

In practice, to calculate the probability of a sentence a start token <s> and an end token </s> are used to indicate the start and end of the sentence respectively.

2.2. Data sparsity problem

For any n-gram that appeared an adequate number of times, we might have a good estimate of its probability. But because any corpus is limited, some perfectly acceptable word sequences are bound to be missing from it. That means, there will be many cases in which correct n-gram sequences will be assigned zero probability. For example, suppose in the training set the bigram একটি(*ekti*) আম(*aam*) occurs 5 times but although correct there is zero occurrence of the similar bigram একটি(*ekti*) আপেল(*apple*). Now suppose we have the following sentence in the test set,

কাদের একটি আপেল খেয়েছে [english] Kader ate an apple
(Kader ekti apple kheychey)

Since the bigram একটি(*ekti*) আপেল(*apple*) has zero count in the training corpus, in the bigram model the probability will be zero as *P*(আপেল(*apple*)|একটি(*ekti*)) = 0. Consequently, the probability of the sentence will be, *P*(কাদের একটি আপেল খেয়েছে) = 0. This probability will be zero since according to (1) the sentence probability is calculated by multiplying the constituent n-gram probabilities and if one of them is zero then total probability will be zero. Therefore, these zero-frequency n-gram sequences that do not occur in the training data but appear in the test set poses great problem for simple n-gram models in accurate probability estimation of the sentences.

2.3. Smoothing

Smoothing techniques are used to keep a LM from assigning zero probability to unseen word sequences, and has become an indispensable part of any LM. In this work, we utilized the two most widely used smoothing algorithms for language modelling namely Witten-Bell (WB) smoothing and Kneser-Ney (KN) smoothing. Smoothing techniques are often implemented in conjunction with two useful strategies that take advantage of the lower order n-grams for the calculation of higher order n-grams that yields zero or low probabilities. These are backoff [24] and interpolation [25] strategies.

2.4. Witten-bell smoothing

Witten Bell (WB) smoothing compensates the counts of word sequences occurring once to estimate the counts of zero frequency word sequences. Originally, WB smoothing algorithm was implemented as a linear interpolation instance taking advantage of lower order n-gram counts.

$$P_{WB-inp}(w_i|w_{i-n+1} \dots w_{i-1}) = \lambda(w_{i-n+1} \dots w_{i-1}) P(w_i|w_{i-n+1} \dots w_{i-1}) + [1 - \lambda(w_{i-n+1} \dots w_{i-1})] p_{WB-interp}(w_i|w_{i-n+2} \dots w_{i-1}) \quad (3)$$

Here, $1 - \lambda(w_{i-n+1} \dots w_{i-1})$ is the total probability mass that is discounted to all the zero n-grams and $\lambda(w_{i-n+1} \dots w_{i-1})$ is the the leftover probability mass of for all non-zero count n-grams. With a little adjustment the WB smoothing can be implemented as an instance of backoff language model. The backoff version of WB smoothing can be written as:

$$P_{WB-bo}(w_i|w_{i-n+1} \dots w_{i-1}) =$$

$$\begin{cases} \lambda(w_{i-n+1} \dots w_{i-1})\, P(w_i|w_{i-n+1} \dots w_{i-1}), if\ c(w_{i-n+1} \dots w_i) > 0 \\ [1 - \lambda(w_{i-n+1} \dots w_{i-1})] P_{WB-bo}(w_i|w_{i-n+2} \dots w_{i-1}), \quad otherwise \end{cases} \tag{4}$$

2.5. Kneser-ney smoothing

In Kneser-Ney (KN) smoothing the lower-order distribution that one combines with a higher-order distribution is built on the intuition that rather than calculating the probability of a word proportional to its number of occurences, it should be calculated based on the number of different words it follows. In its original definition, Kneser and Ney defined KN smoothing as a backoff language model combining lower order models with higher order model using backoff strategy as:

$$p_{KN-bo}(w_i|w_{i-n+1} \dots w_{i-1}) = \begin{cases} \frac{max\{C(w_{i-n+1} \dots w_i) - d, 0\}}{C(w_{i-n+1} \dots w_{i-1})}, if\ c(w_{i-n+1} \dots w_i) > 0 \\ \lambda(w_i|w_{i-n+1} \dots w_{i-1}) p_{KN-bo}(w_i|w_{i-n+2} \dots w_{i-1}), otherwise \end{cases} \tag{5}$$

where $\lambda(w_i|w_{i-n+1} \dots w_{i-1})$ represent the backoff weights assigned to the lower order n-grams which determine the impact of the lower order value on the result. The discount d represents the amount of counts that are discounted from each higher order n-grams. d can be estimated based on the total number of n-grams occurring exactly once (n_1) and twice (n_2) as $d = \frac{n_1}{n_1 + 2n_2}$. The probability for the lower order n-grams can be calculated as

$$p_{KN-bo}(w_i|w_{i-n+2} \dots w_{i-1}) = \frac{T_{1+}(\bullet w_{i-n+2} \dots w_i)}{T_{1+}(\bullet w_{i-n+2} \dots w_{i-1}\bullet)} \tag{6}$$

where, $T_{1+}(\bullet w_{i-n+2} \dots w_i) = |\{w_{i-n+1} : C(w_{i-n+1} \dots w_i) > 0\}|$ and $T_{1+}(\bullet w_{i-n+2} \dots w_{i-1}\bullet) = \sum_{w_i} T_{1+}(\bullet w_{i-n+2} \dots w_i)$. With a little modification the interpolated version KN of can be defined as follows:

$$P_{KN-inp}(w_i|w_{i-n+1} \dots w_{i-1}) = \frac{max\{C(w_{i-n+1} \dots w_i) - d, 0\}}{C(w_{i-n+1} \dots w_{i-1})} + \lambda(w_i|w_{i-n+1} \dots w_{i-1}) P_{KN-inp}(w_i|w_{i-n+2} \dots w_{i-1}) \tag{7}$$

3. PROPOSED GRAMMAR CHEKCING METHODOLOGY

In this section we present the grammar checking methodology that we used to evaluate and analyse the performances of smoothing algorithms. It is an updated version of the grammar checker we presented and described in our previous work. The overall framework or workflow of the system is depicted in Figure 1. The working procedure of the grammar checker consists of three main phases: Training phase, validation phase and testing phase.

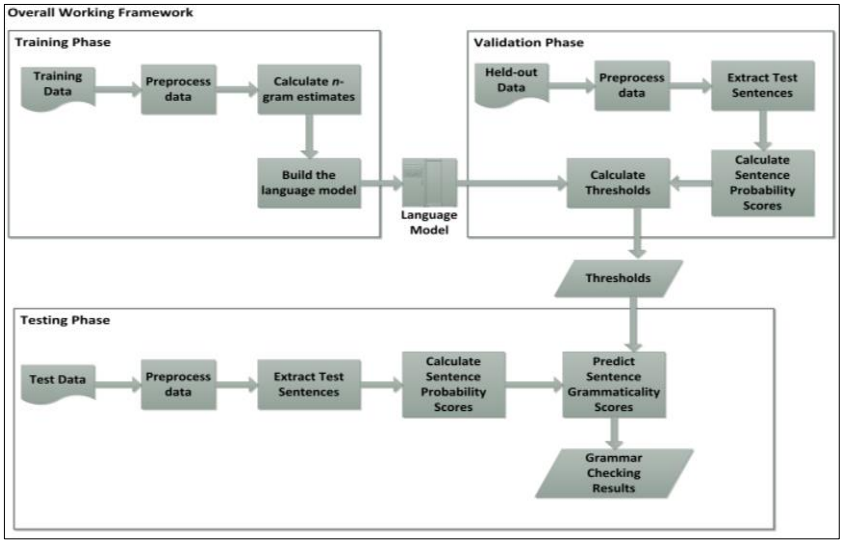

Figure 1. Work flow diagram for proposed the grammar checker

The training process in the proposed system starts by accepting the training corpus and the n value as input. After accepting the input text and the n value, possible n-gram patterns of words are extracted and frequencies of n-grams are then calculated. Using these n-gram frequencies LMs are trained based on the algorithms discussed in the previous sections. In the validation phase, a best possible threshold is calculated for separting the correct or incorrect sentences. The validation process starts by accepting a validation or heldout set consisting of a set of correct and incorrect test sentences. Then the probabilities of these test sentences are calculated and a threshold value is determined that best separates the grammatical and ungrammatical sentences. To do so first we need to define a method to calculate the sentence probability properly which is discussed next.

3.1. Calculation of sentence probability

The sentence probability in n-gram LMs is usually calculated using (1) by first finding the constituent n-grams in the sentence as shown in section 2.1. Since probabilities are by definition less than or equal to 1, the more probabilities we multiply together, the smaller the product becomes. Due to that sentence length (i.e. the number of word tokens in the sentence) has a negative effect on the probability of a sentence. With larger length a sentence tends to have smaller probabilities even though having higher probability constituent n-grams. So, a larger length correct sentence might have smaller probability than a smaller length incorrect sentence because of this effect. To deal with this impact of sentence length on sentence probability calculation a new sentence probability scoring function is introduced in this work defined in (8) by normalizing the sentence probability in (1).

$$P(S) = \sqrt[k]{\prod_{i=1}^{k} P(w_i | w_{i-n+1} \dots w_{i-1})} \quad (8)$$

3.2. Optimal threshold calculation

In the validation phase, optimal threshold for the n-gram based classfier is calculated in two stages. In the first stage, we used 10-fold cross validation on the training set which consists of only grammatically correct senetcnes. Since, a correct sentence typically has a higher probability than an incorrect one, in each fold we selected the lowest probability score among the sentences of training part as the threshold and used that threshold to classify the test sentences and find the misclassification error with that threshold. The threshold that has the minimum misclassifcaton error is finally chosen as the final threshold. The process is an improved version to the process we used in our previous work. The process is explained in Algorithm 1.

Algorithm 1. Priliminary threshold selection from training set in stage 1

Input: *S*= training data set*;*
L = corresponding true labels of positive and negative sentences in VS
LM = language model to be used
1. Divide the data set into 10 equal sized subsets as $S = \{S_1, S_2, \dots, S_{10}\}$
2. Set MCR_{min}= 1 //the minimum misclassification rate and
   Set *T* = final threshold
3. For i = 1 to 10 Do,
4. Set $S_{test} = S_i$ and $S_{train} = S - S_i$
5. Train the *LM* on S_{train}.
6. *t* = Find the minimum probability in S_{train} and set it as current threshold
7. *probs* = Test the LM on S_{test} *using t as threshold*..
8. *mcr* = Find the misclassification rate for the current threshold.
9. If mcr < MCR_{min} *then Set* MCR_{min} *= mcr and T= t*
10. End For
11. return *T //T* is the final threshold selected

Though methods in the first stage work well but they introduce a lot of false positives in the final classification. Since we are using the minimum probability score of correct or positive sentences as threshold it ensures high true positives but it adversely overlaps with a substantial number of incorrect sentences in the probability distribution. Hence, the high false positive rates. To reduce the unwanted high number of false positives and to improve the classification performance overall in the second stage we used a method that gradually increases the threshold to reduce the number of false positives but also ensures the balance between false positives and false negatives. This method is applied on a separate validation set consisting of equal number of positive and negative sentences to finalize the optimal threshold. This process is explained in Algorithm 2.

In the testing phase, the classification LMs are tested on a separate test set consisting of grammatical and ungrammatical sentences using the optimum threshold calculated in the validation phase. If any senetence has a probability less than the optimum threshold then it is classified as ungrammatical otherwise grammatical.

Algorithm 2. Optimal threshold selection from validation set in stage 2

Input: t_0 = preliminary threshold calculated from the training set using Algorithm 1*;*
VS = validation set
L = corresponding true labels of positive and negative sentences in VS
LM = language model to be used

1. Calculate [*TP,FP,TN,FN*] using t_0 as threshold testing on *VS* where *TP* = no. of true positives, *FP* = no. of false positives, *TN* = no. of true negatives, *FN* = no. of true positives
   Calculate *FPR = FP/(TN+FP), FNR =FN/(TP+FN) and MCR=FPR+FNR where FPR = false positive rate, FNR = false negative rate and MCR = overall misclassification rate*
2. Set *th* = t_0 // *th* is the final threshold
   Divide the range [t_0, 1] into *k* equal sized thresholds in THS = { $t_1, t_2, ..., t_k$}
3. For each threshold *t* in THS Do,
4. Calculate [*TP,FP,TN,FN*] using *t* as threshold on *VS* and hence calculate the fpr_t and fnr_t for *t.*
5. If $fpr_t \leq fnr_t$ *and MCR* $\geq fpr_t + fnr_t$ then,
6. Set *th = t, FPR =* fpr_t*, FNR =* fnr_t *and MCR = FPR+FNR*
7. End If
8. End For
9 return *th //th* is the final threshold selected

4. RESULTS AND ANALYSIS

The main focus of this section is to investigate the performance of the grammar checking system based on certain factors such as the smoothing algorithm used, n-gram orders, length of the target sentences etc. To train and test the LMs we used a large corpus of 20 million words containing 181820 grammatically correct sentences. Around 80% of the corpus is used for training purpose. The validation set consists of 20000 correct sentences and 20000 incorrect sentences. The grammatically incorrect sentences are artificially created by inseting, deleting or replacing words in the correct sentences in the set. The test set contains 15000 correct and 15000 incorrect sentences. In our previous work, we only tested the methods on a test set containing only simple senetnces of length of 5-10 words. This time we tested the techniques on a more difficult and practical test set consisting of all kinds of simple, complex and compound sentences with lengths ranging from 5 to 20 words. The experiments have been tested on a machine with 2.40GHz Intel Core i3 processor and 12 GB of RAM, running on Microsoft Windows 8. The experimental system has been developed using python programming language. The comparative performances of the LMs were evaluated by precision, recall and f-scores. The overall performances of the different LMs based on the smooting techniques and n-gram order used are presented in Table 1.

Table 1 represents the results of different LMs for each metric (precision, recall & f-score) in two columns. The gray shaded column represents the results obtained using the threshold selection method used in our previous work [22] and the other column represents the results attained using our two stage threshold selection procedure explained in Algorithm 1 and Algorithm 2, which is proposed in this work. Our newly proposed two stage optimum threshold selection approach clearly provides significantly improved results for all the LMs compared to the previous approach. It significantly increases the precision and hence the overall f-score for all the LMs with the cost of small or insignificant reduction in recall values for grammatical sentences. Similarly, for ungrammatical sentences the recall scores are significantly improved resulting in much improved f-score with the negligible loss of precision values. This improved performance is due to the reduction in false positives and also keeping a balance between false positives and false negatives. These results prove the superiority of our proposed method compared to the previous one. From the newly found results in Table 1 it is evident that, KN-interp with its 5-gram model clearly outperforms all the other LMs in terms of precision, recall and f-score for both grammatical and ungrammatical sentences achieving highest f-scores of 72.92% and 68.51% respectively. In terms of f-score as we can see from the Table 1, WB-backoff produces the second best results for both grammatical and ungrammatical sentences with KN-backoff model providing the third best performance. The models rank similiarly in terms of precision and recall with one or two exceptions such as for recall metric KN-backoff performs slightly better than WB-backoff.

Table 1. Performances of different LMs

Models	N-gram Order	Performances with Grammatical Data						Performances with Ungrammatical Data					
		Precision		Recall		F-score		Precision		Recall		F-score	
		With *T1*	With *T2*	With *T1*	With *T2*	With *T1*	With *T2*	With *T1*	With *T2*	With *T1*	With *T2*	With *T1*	With *T2*
WB-backoff	2	34.56%	42.10%	52.67%	51.54%	41.74%	46.35%	38.69%	37.53%	24.89%	29.11%	30.29%	32.79%
	3	52.31%	61.81%	73.36%	71.57%	61.07%	66.33%	68.29%	66.24%	50.76%	55.78%	58.24%	60.56%
	4	55.43%	66.48%	75.76%	73.55%	64.02%	69.84%	72.58%	70.40%	56.32%	62.91%	63.43%	66.45%
	5	56.31%	66.42%	75.29%	74.25%	64.43%	70.12%	73.01%	70.81%	55.89%	62.46%	63.31%	66.37%
WB-interp	2	31.87%	40.45%	51.51%	50.55%	39.38%	44.94%	35.15%	34.09%	20.32%	25.58%	25.75%	29.23%
	3	52.12%	60.70%	69.47%	67.91%	59.56%	64.10%	65.55%	63.58%	47.98%	56.02%	55.41%	59.56%
	4	53.11%	64.38%	73.85%	72.26%	61.79%	68.10%	70.52%	68.40%	50.21%	60.03%	58.66%	63.94%
	5	55.02%	64.70%	75.34%	73.36%	63.60%	68.76%	71.39%	69.24%	53.41%	59.97%	61.10%	64.27%
KN-backoff	2	32.12%	38.81%	50.12%	48.61%	39.15%	43.16%	32.22%	31.25%	19.97%	23.36%	24.66%	26.73%
	3	49.18%	59.64%	69.92%	68.02%	57.75%	63.56%	64.75%	62.80%	47.65%	53.97%	54.90%	58.05%
	4	51.55%	62.61%	76.56%	74.84%	61.61%	68.18%	70.86%	68.73%	50.33%	55.31%	58.86%	61.29%
	5	52.44%	64.01%	78.93%	77.00%	63.01%	69.91%	73.35%	71.14%	50.91%	56.71%	60.10%	63.11%
KN-interp	2	35.76%	44.79%	56.30%	55.36%	43.74%	49.52%	42.86%	41.57%	25.87%	31.76%	32.26%	36.01%
	3	52.89%	62.61%	74.60%	72.99%	61.90%	67.40%	69.72%	67.62%	48.79%	56.42%	57.41%	61.52%
	4	57.09%	67.18%	77.38%	76.46%	65.70%	71.52%	73.63%	72.69%	55.88%	62.64%	63.54%	67.29%
	5	58.71%	**68.15%**	79.51%	**78.41%**	67.54%	**72.92%**	75.70%	**74.58%**	56.10%	**63.35%**	64.44%	**68.51%**

*Here, *T1* is the threshold calculated using the threshold selection algorithm defined in our previous work [22]. *T2* is the threshold calculated using the two-stage threshold selection technique introduced in this work.

Performances of the LMs inprove with the growing order of *n*-gram and the performance improvement gets lesser with each higher order. Though the performances of most of the LMs tend to increase from 4-gram order to 5-gram order, the performance differences are very insignificant. Figure 2 and Figure 3 depict this effect where the f-scores of the LMs varied by the *n*-gram order are presented for both grammatical and ungrammatical sentences. Though not presented here, similar effects can be observed in terms of precision and recall.

Since we are using a data set consisting of varied length of sentences, next we try to find out whether sentence length has any effect on the performances of LMs. Figures 4 and 5 present the f-scores of two of our best performing LMs, KN-interp and WB-backoff varied by the length of sentences tested for both grammatical and ungrammatical data respectively. From Figures 4 and 5, we find that the performances of the LMs gradually decrease with the increasing sentence length for the sentences. This is understandable since sentences with more words or higher length will tend to be more complex in structure and difficult to be judged. But this degradation in performance is linear not exponential and changes are very small. This shows the robustness of our sentence probability calculation function defined in (8). Though not presented here, performances of other LMs (KN-backoff and WB-interp) and on other metrics shows similar characteristics for the dependency of the method on sentence length. So, we can conclude that KN LM with its interpolated version i.e. KN-interp outperforms all the other LMs in terms of all performance metrics. With higher *n*-gram order the performances of the LMs improve with 4-gram and 5-gram models showing similar performances with negligible diffrences and the length of the sentence does not affect the performance of the LMs significantly.

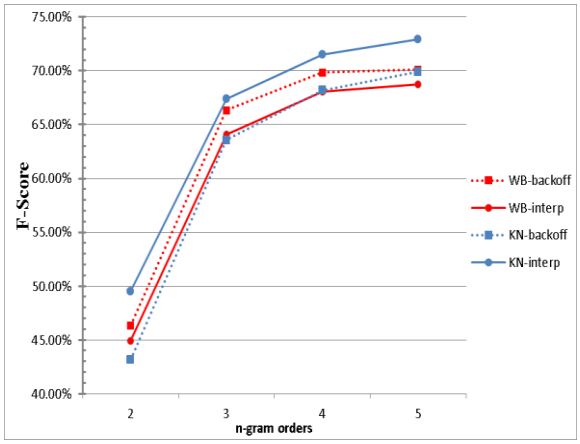

Figure 2. Effect of *n*-gram order on the performances of LMs for grammatical data

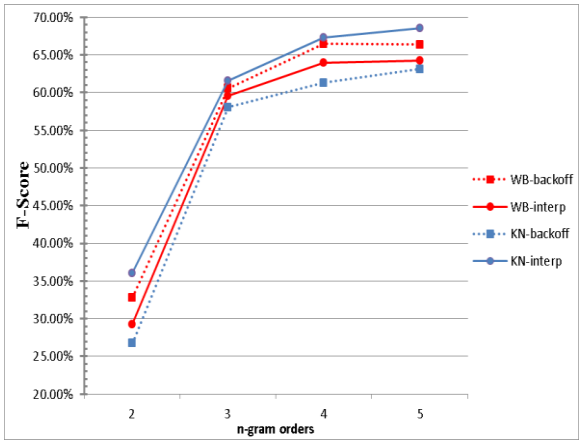

Figure 3. Effect of *n*-gram order on the performances of LMs for ungrammatical data

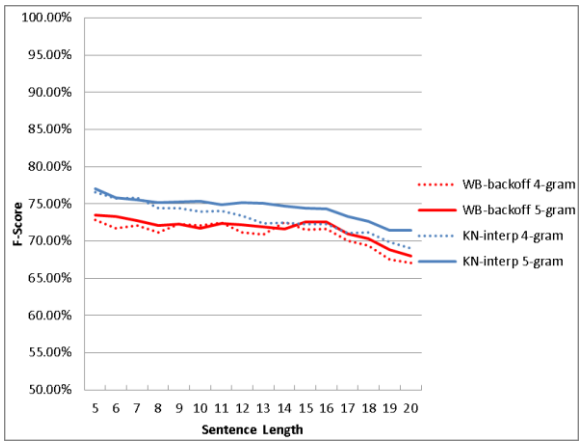

Figure 4. Effect of sentence length on the performances of LMs for grammatical data

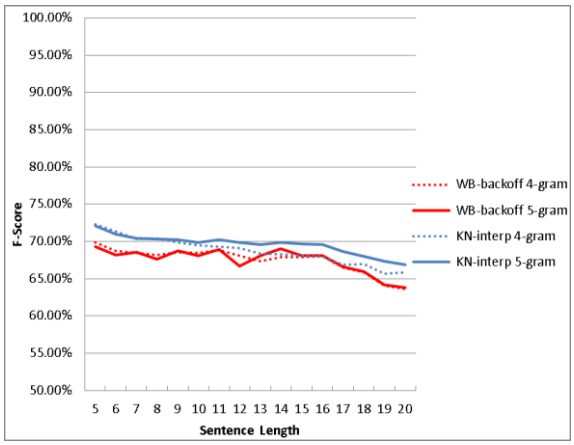

Figure 5. Effect of sentence length on the performances of LMs for ungrammatical data

5. CONCLUSION

The goal of this research was to design and develop a robust grammar checking system for Bangla language which can accurately judge realistic, simple and complex sentences for grammaticality. To attain that extent, a statistical grammar checking system based on n-gram language modelling has been designed and developed. To achieve robust performance with n-gram models two most widely used smoothing techniques namely Kneser-ney and Witten-bell were used and compared to find best performing system. Furthermore, the LMs' performances were tested on a newly developed challenging test set containing 30000 all types of simple, complex and compound sentences to attain realistic performance results. Our experimental results show that Kneser-ney interpolated smoothing based 5-gram LM outperforms others in terms of all the metrics achieving f-scores of 72.92% and 68.51% for grammatical and ungrammatical data respectively. For further this research work, more features such as parts of speech tags and other linguistic features can be added to improve the performance of the system.

REFERENCES

[1] E. D. Liddy, "Natural language processing in Encyclopaedia of Library and Information Science," 2nd Edition, Florida: CRC Press, pp. 1-20, 2001.
[2] Shafaf Ibrahim, Nurul Amirah Zulkifli, Nurbaity Sabri, Anis Amilah Shari, and Mohd Rahmat Mohd Noordin, "Rice grain classification using multi-class support vector machine (SVM)," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 8, no. 3, pp. 215-220, Sep. 2019.
[3] Amirul Sadikin Md Affendi, Marina Yusoff, "Review of anomalous sound event detection approaches," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 8, no. 3, pp. 264-269, Sep. 2019.
[4] Cesar G. Pachon-Suescun, Carlos J. Enciso-Aragon, Robinson Jimenez-Moreno, "Robotic Navigation Algorithm with Machine Vision," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 2, pp. 1308-1316, Apr. 2020.
[5] K.A.F.A. Samah, I.M. Badarudin, E.E. Odzaly, K.N. Ismail, N.I.S. Nasarudin, N.F. Tahar, M.H. Khairuddin, "Optimization of house purchase recommendation system (HPRS) using genetic algorithm," *Indonesian Journal of Electrical Engineering and Computer Science* (IJEECS), vol. 16, no. 3, pp. 1530-1538, Dec. 2019.
[6] Vernon A., "Computerized grammar checkers 2000: Capabilities, limitations, and pedagogical possibilities," *Computers and Composition*, vol. 17, no. 3, pp. 329-49, Dec. 2000.
[7] Richardson, S., "Microsoft natural language understanding system and grammar checker," In *Fifth Conference on Applied Natural Language Processing: Descriptions of System Demonstrations and Videos*, pp. 20-20, 1997
[8] Arppe A., "Developing a grammar checker for Swedish," In *Proceedings of the 12th Nordic Conference of Computational Linguistics (NODALIDA 1999)*, pp. 13-27, 2000.
[9] Shaalan KF., "Arabic GramCheck: A grammar checker for Arabic," *Software: Practice and Experience*, vol. 35, no. 7, pp. 643-65, Jun. 2005
[10] Bopche L, Dhopavkar G, Kshirsagar M., "Grammar Checking System Using Rule Based Morphological Process for an Indian Language," In *Global Trends in Information Systems and Software Applications*, Springer, Berlin, Heidelberg, pp. 524-531, 2012.
[11] Jensen K, Heidorn GE, Richardson SD, "Natural language processing: the PLNLP approach," *Springer Science & Business Media*, Dec. 2012.
[12] Manning CD, Raghavan P, Schutze H., "Introduction to Information Retrieval Ⅱ," Cambridge University Press, Ch. 20, pp. 405-416, 2008.

[13] Martin JH, Jurafsky D., "Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition," Pearson/Prentice Hall, 2009.
[14] Witten IH, Bell TC., "The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression," *IEEE transactions on information theory*, vol. 37, no. 4, pp. 1085-94, Jul. 1991.
[15] Kneser R, Ney H., "Improved backing-off for m-gram language modeling. In Acoustics, Speech, and Signal Processing," *1995 International Conference on Acoustics, Speech, and Signal Processing ICASSP-95, IEEE*, vol. 1, pp. 181-184, May 1995.
[16] Md. Iftakher Alam Eyamin, Md. Tarek Habib, M. Ifte Khairul Islam, Md. Sadekur Rahman, Md. Abbas Ali Khan, "An Investigative Design of Optimum Stochastic Language Model for Bangla Autocomplete," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 13, no. 2, pp. 671-676, 2019.
[17] Muhammad Ifte Khairul Islam, Md. Tarek Habib, Md. Sadekur Rahman and Md. Riazur Rahman, "A Context-Sensitive Approach to Find Optimum Language Model for Automatic Bangla Spelling Correction," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 11, pp. 184-191, 2018.
[18] Md. Tarek Habib, Abdullah Al-Mamun, Md. Sadekur Rahman, Shah Md. Tanvir Siddiquee and Farruk Ahmed, "An Exploratory Approach to Find a Novel Metric Based Optimum Language Model for Automatic Bangla Word Prediction," *International Journal of Intelligent Systems and Applications (IJISA)*, vol. 10, no. 2, pp. 47-54, 2018.
[19] J. Lane, "The 10 Most Spoken Languages in the World," *Babbel Magazine*, 2019. [Online], Available: https://www.babbel.com/en/magazine/the-10-most-spoken-languages-in-the-world, [Accessed: 18 March 2018].
[20] Alam M. Jahangir, Naushad UzZaman, and Mumit Khan, "N-gram based Statistical Grammar Checker for Bangla and English," *In Proceeding of ninth International Conference on Computer and Information Technology (ICCIT 2006)*, 2006.
[21] Nur Hossain Khan M, Khan F, Islam MM, Rahman MH, Sarke B., "Verification of Bangla Sentence Structure using N-Gram," *Global Journal of Computer Science and Technology*, May 2014.
[22] Rahman MR, Habib MT, Rahman MS, Shuvo SB, Uddin MS., "An Investigative Design Based Statistical Approach for Determining Bangla Sentence Validity," *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 16, no. 11, pp. 30, Nov. 2016.
[23] Charniak E., "Statistical language learning," MIT press, 1996.
[24] Katz S., "Estimation of probabilities from sparse data for the language model component of a speech recognizer," *IEEE transactions on acoustics, speech, and signal processing*, vol. 35, no. 3, pp. 400-1, Mar. 1987.
[25] Jelinek F., "Interpolated estimation of Markov source parameters from sparse data," In *Proc. Workshop on Pattern Recognition in Practice*, 1980.