BSpell: A CNN-Blended BERT Based Bangla Spell Checker
Chowdhury Rafeed Rahman
National University of Singapore
e0823054@u.nus.edu

MD. Hasibur Rahman
United International University

Samiha Zakir and Mohammed Rafsan
University of Texas Rio Grande Valley

Mohammed Eunus Ali
Bangladesh University of Engineering and Technology

Abstract
Bangla typing is mostly performed using English keyboard and can be highly erroneous due to the presence of compound and similarly pronounced letters. Spelling correction of a misspelled word requires understanding of word typing pattern as well as the context of the word usage. A specialized BERT model named BSpell has been proposed in this paper targeted towards word for word correction in sentence level. BSpell contains an end-to-end trainable CNN sub-model named SemanticNet along with specialized auxiliary loss. This allows BSpell to specialize in highly inflected Bangla vocabulary in the presence of spelling errors. Furthermore, a hybrid pretraining scheme has been proposed for BSpell that combines word level and character level masking. Comparison on two Bangla and one Hindi spelling correction dataset shows the superiority of our proposed approach. BSpell is available as a Bangla spell checking tool via GitHub: https://github.com/Hasiburshanto/Bangla-Spell-Checker.

1 Introduction
Bangla is the native language of 228 million people which makes it the sixth most spoken language in the world 1. This Sanskrit originated language has 11 vowels, 39 consonants, 11 modified vowels and 170 compound characters (Sifat et al., 2020). There is vast difference between Bangla grapheme representation and phonetic utterance for many commonly used words. As a result, fast typing of Bangla yields frequent spelling mistakes. Almost all Bangla native speakers type using English QWERTY layout keyboard (Noyes, 1983) which makes it difficult to type Bangla compound characters, phonetically similar single characters and similar pronounced modified vowels correctly. Thus Bangla typing speed, if error-free typing is desired, is slow. An accurate spell checker (SC) can be a solution to this problem.

Existing Bangla SCs include phonetic rule (Uz-Zaman and Khan, 2004, 2005) and clustering based methods (Mandal and Hossain, 2017). These methods do not take misspelled word context into consideration. Another N-gram based Bangla SC (Khan et al., 2014) takes only short range previous context into consideration. Recent state-of-the-art (SOTA) spell checkers have been developed for Chinese language, where a character level confusion set (similar characters) guided sequence to sequence (seq2seq) model has been proposed by Wang et al. (2019). Another research used similarity mapping graph convolutional network in order to guide BERT based character by character parallel correction (Cheng et al., 2020). Both these methods require external knowledge and assumption about confusing character pairs existing in the language. The most recent Chinese SC offers an assumption free BERT architecture where error detection network based soft-masking is included (Zhang et al., 2020). This model takes all N characters of a sentence as input and produces the correct version of these N characters as output in a parallel manner.

| Incorrect | Correct |
|-----------|---------|
| পিরকা (প+র+ি◌+ক+◌া) | পরীা (প+র+ি◌ী+ক+◌্+ষ+◌া): Exam |
| িবশশ (ব+ি◌+শ+শ) | িব (ব+ি◌+শ): World |
| ভাদর (ভ+ঔ+দ+র) | ভার (ভ+ঔ+দ+র): month name |

Figure 1: Heterogeneous character number between error word and corresponding correctly spelled word

One of the limitations in developing Bangla SC using SOTA BERT based implementation (Zhang et al., 2020) is that number of input and output characters in BERT has to be exactly the same. Such scheme is only capable of correcting substitution type errors. As compound characters are common in Bangla words, an error made due to the
substitution of such characters also changes word length (see the table in Figure 1). So, we introduce word level prediction in our proposed BERT based model.

| Correct | Incorrect |
|---------|-----------|
| সৈনিক যোদ্ধা ছড়া ফুলু পেল। | সৈনিক যোদ্ধা ছড়া ফুলু পেল। |
| অবসর হওনা করো না। | অবসর হওনা করো না। |
| কাল আমাদের রক্ষার পার্থক্য। | কাল আমাদের রক্ষার পার্থক্য। |

Figure 2: ample words that are correctly spelled accidentally, but are context-wise incorrect.

The table shown in Figure 2 illustrates the importance of context in Bangla SC. Although the red marked words of this figure are the misspelled versions of the corresponding green marked correct words, these red words are valid Bangla words. But if we check these red words based on sentence semantic context, we can realize that these words have been produced accidentally because of spelling error. An effective SC has to consider word pattern, its prior context and its post context.

| Misspelled: পাল জিনিসের আমার বিভক্তিক্ষণিলি | Correct: প্রচ্ছায় ওঝর বিভক্তিক্ষণিলি |
|---------------------------------|---------------------------------|
| Meaning: Villages are dependent on agriculture |

Figure 3: Necessity of understanding existing erroneous words for spelling correction of misspelled words

Spelling errors often span up to multiple words in a sentence. Figure 3 provides an example where all four words have been misspelled. The correction of each word has context dependency on a few other words of the same sentence. The problem is that these words that form the correction context are also misspelled. The table in the figure shows the words to look at in order to correct each misspelled word. In the original sentence (colored in red), all these words that need to be looked at for context are misspelled. If a SC cannot understand the approximate underlying meaning of these misspelled words, then we lose all context for correcting each misspelled word which is undesirable.

We propose a word level BERT (Devlin et al., 2018) based model BSpell. This model is capable of learning prior and post context dependency through the use of multi-head attention mechanism of stacked Transformer encoders (Vaswani et al., 2017). The model uses CNN based learnable SemanticNet sub-model to capture semantic meaning of both correct and misspelled words. BSpell also uses specialized auxiliary loss to facilitate word level pattern learning and vanishing gradient problem removal. We introduce hybrid pretraining for BSpell to capture both context and word error pattern. We perform detailed evaluation on three error datasets that include a real life Bangla error dataset. Our evaluation includes detailed analysis on possible LSTM based SCs, SC variants of BERT and existing classic Bangla SCs.

2 Related Works

Several studies on Bangla SC development have been conducted in spite of Bangla being a low resource language. A phonetic encoding oriented Bangla word level SC based on Soundex algorithm was proposed by UzZaman and Khan (2004). This encoding scheme was later modified to develop a Double Metaphone encoding based Bangla SC (UzZaman and Khan, 2005). They took into account major context-sensitive rules and consonant clusters while performing their encoding scheme. Another word level Bangla SC able to handle both typographical and phonetic errors was proposed by Mandal and Hossain (2017). An N gram model was proposed by Khan et al. (2014) for checking sentence level Bangla word correctness. An encoder-decoder based seq2seq model was proposed by Islam et al. (2018) for Bangla sentence correction task which involved bad arrangement of words and missing words, though this work did not include incorrect spelling. A recent study has included Hindi and Telugu SC development, where mistakes are assumed to be made at character level (Etoori et al., 2018). They have used attention based encoder-decoder modeling as their approach.

SOTA research in this domain involves Chinese SCs as it is an error prone language due to its confusing word segmentation, phonetically and visually similar but semantically different characters. A seq2seq model assisted by a pointer network was employed for character level spell checking where the network is guided by externally generated character confusion set (Wang et al., 2019). Another research incorporated phonological and visual similarity knowledge of Chinese characters into BERT based SC model by utilizing graph
convolutional network (Cheng et al., 2020). A recent BERT based SC has taken advantage of GRU (Gated Recurrent Unit) based soft masking mechanism and has achieved SOTA performance in Chinese character level SC in spite of not providing any external knowledge to the network (Zhang et al., 2020). Another external knowledge free approach namely FASPell used BERT based seq2seq model (Hong et al., 2019). HanSpeller++ is notable among initially implemented Chinese SCs (Xiong et al., 2015). It was an unified framework utilizing a hidden Markov model.

3 Our Approach

3.1 Problem Statement

Suppose, an input sentence consists of \( n \) words – \( Word_1, Word_2, \ldots, Word_n \). For each \( Word_i \), we have to predict the right spelling, if \( Word_i \) exists in the top-word list of our corpus. If \( Word_i \) is a rare word (Proper Noun in most cases), we predict \( UNK \) token denoting that we do not make any correction to such words. For correcting a particular \( Word_i \) in a paragraph, we only consider other words of the same sentence for context information.

3.2 BSpell Architecture

Figure 4 shows the details of BSpell architecture. Each input word of the sentence is passed through the SemanticNet sub-model. This sub-model returns us with a SemanticVec vector representation for each input word. These vectors are then passed onto two separate branches (main branch and secondary branch) simultaneously. The main branch is similar to BERT_Base architecture (Gong et al., 2019). This branch provides us with the \( n \) correct words corresponding to the \( n \) input sentence words at its output side. The secondary branch consists of an output dense layer. This branch is used for the sole purpose of imposing auxiliary loss to facilitate SemanticNet sub-model learning of misspelled word patterns.

3.2.1 SemanticNet Sub-Model

Correcting a particular word requires the understanding of other relevant words in the same sentence. Unfortunately, those relevant words may also be misspelled. As humans, we can understand the meaning of a word even if it is misspelled because of our deep understanding at word syllable level and our knowledge of usual spelling error pattern. We want our model to have similar semantic level understanding of the words. We propose SemanticNet, a sequential 1D CNN sub-model that is employed at each individual word level with a view to learning intra word syllable pattern. Details of individual word representation has been shown in the bottom right corner of Figure 4. We represent each input word by a matrix (each character represented as a one hot vector). We apply global max pooling on the final convolution layer output feature matrix of SemanticNet which gives us the SemanticVec vector representation of the input word. We get a similar SemanticVec representation from each of our input words by independently applying the same SemanticNet sub-model on each of their matrix representations.

3.2.2 BERT_Base as Main Branch

Each of the SemanticVec vector representations obtained from the input words are passed parallely on to our first Transformer encoder. 12 such Transformer encoders are stacked on top of each other. Each Transformer employs multi head attention mechanism, layer normalization and dense layer specific modification on each input vector. The attention mechanism applied on the word feature vectors in each transformer layer helps the words of the input sentence interact with one another extracting sentence context. We pass the final Transformer layer output vectors to a dense layer with Softmax activation function applied on each vector in an independent manner. So, now we have \( n \) probability vectors from \( n \) words of the input sentence. Each probability vector contains \( len_P \) values, where \( len_P \) is one more than the total number of top words considered (the additional word represents rare words). The top word corresponding to the index of the maximum probability value of \( i^{th} \) probability vector represents the correct word for \( Word_i \) of the input sentence.

3.2.3 Auxiliary Loss in Secondary Branch

Gradient vanishing problem is a common phenomena in deep neural networks, where weights of the shallow layers are not updated sufficiently during backpropagation. With the presence of 12 Transformer encoders on top of the SemanticNet sub-model, the layers of this sub-model certainly lie in a shallow position. Although SemanticNet constitutes a small initial portion of BSpell, this portion is responsible for word pattern learning, an important task of SC. In order to eliminate gradient vanishing problem of SemanticNet and to turn it into an ef-
effective pattern based word level spell checker, we introduce an auxiliary loss based secondary branch in BSpell. Each of the $n$ SemanticVecs obtained from the $n$ input words are passed parallelly on to a Softmax layer without any further modification. The outputs obtained from this branch are probability vectors similar to the main branch output. The total loss of BSpell can be expressed as: $L_{Total} = L_{Final} + \lambda \times L_{Auxiliary}$. We want our final loss to have greater impact on model weight update as it is associated with the final prediction made by BSpell. Hence, we impose the constraint $0 < \lambda < 1$. This secondary branch of BSpell does not have any Transformer encoders through which the input words can interact to produce context information. The prediction made from this branch is dependent solely on misspelled word pattern extracted by SemanticNet. This enables SemanticNet to learn more meaningful word representation.

### 3.3 BERT Hybrid Pretraining

In contemporary BERT pretraining methods, each input word $Word_i$ maybe kept intact or maybe replaced by a default mask word in a probabilistic manner (Devlin et al., 2018; Liu et al., 2019). BERT has to predict the masked words. Mistakes from the BERT side will contribute to loss value accelerating backpropagation based weight update. In this process, BERT learns to fill in the gaps, which in turn teaches the model language context.
Sun et al. (2020) proposed incremental ways of pretraining the model for new NLP tasks. We take a more task specific approach for masking. In SC, recognizing noisy word pattern is important. But there is no provision for that in contemporary pretraining schemes and so, we propose hybrid masking (see Figure 5). Among \( n \) input words in a sentence, we randomly replace \( n_W \) words with a mask word \( Mask_W \). Among the remaining \( n - n_W \) words, we choose \( n_C \) words for character masking. We choose \( m_C \) characters at random from a word having \( m \) characters to be replaced by a mask character \( Mask_C \) during character masking. Such masked characters introduce noise in words and helps BERT to understand the probable semantic meaning of noisy/misspelled words.

4 Experimental Setup
4.1 Implemented Pretraining Schemes
We have experimented with three types of masking based pretraining schemes. During word masking we randomly select 15% words of a sentence and replace those with a fixed mask word. During character masking, we randomly select 50% words of a sentence. For each selected word, we randomly mask 30% of its characters by replacing each of them with a special mask character. Finally, during hybrid masking, we randomly select 15% words of a sentence and replace them with a fixed mask word. We randomly select 40% words from the remaining words. For these selected words, we randomly mask 25% of their characters.

4.2 Dataset Specification
We have used one Bangla and one Hindi corpus with over 5 million (5 M) sentences for BERT pretraining (see Table 1). Bangla pretraining corpus consists of Prothom Alo 2 articles dated from 2014-2017 and BDNews24 3 articles dated from 2015-2017. The Hindi pretraining corpus consists of Hindi Oscar Corpus 4, preprocessed Wikipedia articles 5, HindiEnCorp05 dataset 6 and WMT Hindi News Crawl data 7 (all of these are publicly available corpus). We have used Prothom-Alo 2017 online newspaper dataset for Bangla SC training and validation purpose. Our errors in this corpus have been produced synthetically using the probabilistic algorithm described by Sifat et al. (2020). We further validate our baselines and proposed methods on Hindi open source SC dataset, namely Tools-ForIL (Etoori et al., 2018). For real error dataset, we have collected a total of 6300 sentences from Nayadiganta 8 online newspaper. Then we have distributed the dataset among ten participants. They have typed (in regular speed) each correct sentence using English QWERTY keyboard producing natural spelling errors. It has taken 40 days to finish the labeling. Top words have been taken such that they cover at least 95% of the corresponding corpus.

4.3 BSpell Architecture Hyperparameters
SemanticNet sub-model of BSpell consists of a character level embedding layer producing a 40 size vector from each character, then 5 consecutive layers each consisting of 1D convolution (batch normalization and Relu activation in between each pair of convolution layers) and finally, a 1D global max pooling in order to obtain SemanticVec representation from each input word. The five 1D convolution layers consist of \((64, 2), (64, 3), (128, 3), (128, 3), (256, 4)\) convolution, respectively. The first and second element of each tuple denote number of convolution filters and kernel size, respectively. We provide a weight of 0.3 (\(\lambda\) value of loss function) to the auxiliary loss. The main branch of BSpell is similar to BERT_Base (Gong et al., 2019) in terms of stacking 12 Transformer encoders. Attention outputs from each Transformer is passed through a dropout layer (Srivastava et al., 2014) with a dropout rate of 0.3 and then layer normalized (Ba et al., 2016). We use Stochastic Gradient Descent (SGD) Optimizer with a learning rate of 0.001 for our model weight update. We clip our gradient value and keep it below 5.0 to avoid gradient exploding problem.

\(^2\)https://www.prothomaloi.com/
\(^3\)https://bangla.bdnews24.com/
\(^4\)https://www.kaggle.com/abhishek/hindi-oscar-corpus
\(^5\)https://www.kaggle.com/disisbig/hindi-wikipedia-articles-172k
\(^6\)http://hdl.handle.net/11858/00-097C-0000-0023-625F-0
\(^7\)https://www.aclweb.org/anthology/W19-5301
\(^8\)https://www.dailynayadiganta.com/
5 Results and Discussion

5.1 Training and Validation Details
In case of Bangla SC, we randomly initialize the weights of model \( M \). We use our large Bangla pretrain corpus for hybrid pretraining and get pre-trained model \( M_{pre} \). Next we split our benchmark synthetic spelling error dataset (Prothom-Alo) into 80%-20% training-validation set. We fine tune \( M_{pre} \) using the 80% training portion (obtaining fine tuned model \( M_{fine} \)) and report performance on the remaining 20% validation portion. We use the Bangla real spelling error dataset in two ways - (1) We do not fine tune \( M_{fine} \) on any of part of this data and use the entire dataset as an independent test set (result reported with the title real error (no fine tune)) (2) We split this real error dataset into 80%-20% training-validation and fine tune \( M_{fine} \) further using the 80% portion, then validate on the remaining 20% (result reported with the title real error (fine tuned)). In case of Hindi, the first two steps (pretraining and fine tuning) are the same. We have not constructed any real life spelling error dataset for Hindi. So, results are reported on the 20% held out portion of the benchmark dataset.

5.2 BSpell vs Contemporary BERT Variants
We start with BERT Seq2seq where the encoder and decoder portion consist of 12 stacked Transformers (Devlin et al., 2018). Predictions are made at character level. Similar architecture has been used in FASpell (Hong et al., 2019) for Chinese SC. A word is considered wrong if even one of its characters is predicted incorrectly. Hence character level seq2seq modeling achieves poor result (see Table 2). Moreover, in most cases during sentence level spell checking, the correct spelling of the \( i^{th} \) word of input sentence has to be the \( i^{th} \) word in the output sentence as well. Such constraint is difficult to follow through such architecture design. BERT Base consisting of stacked Transformer encoders has two differences from the design proposed by Cheng et al. (2020) - (i) We make predictions at word level instead of character level (ii) We do not incorporate any external knowledge about Bangla SC since such knowledge is not well established in the field. This approach achieves good performance in all four cases. Soft Masked BERT learns to apply specialized synthetic masking on error prone words in order to push the error correction performance of BERT Base further. The error prone words are detected using a GRU sub-model and the whole architecture is trained end to end. Although Zhang et al. (2020) implemented this architecture to make corrections at character level, our implementation does everything in word level. We have used popular FastText (Athiwaratkun et al., 2018) word representation for both BERT Base and Soft Masked BERT. BSpell shows decent performance improvement in all cases.

5.3 Comparing BSpell Pretraining Schemes
We have implemented three different pretraining schemes (details provided in Subsection 4.1) on BSpell before fine tuning on spell checker dataset. Word masking teaches BSpell context of a lan-

| Datasets                                | Unique Word | Unique Char | Top Word | Train Sample | Validation Sample | Unique Error Word | Error Word Percentage |
|-----------------------------------------|-------------|-------------|----------|--------------|-------------------|-------------------|-----------------------|
| Prothom-Alo Bangla Synthetic Error      | 262 K       | 73          | 35 K     | 1 M          | 200 K             | 450 K             | 52%                   |
| Bangla Real Error                       | 14.5 K      | 73          | _        | 4.3 K        | 2 K               | 10 K              | 36%                   |
| Bangla Pretrain Corpus                  | 513 K       | 73          | 40 K     | 5.5 M        | _                 | _                 | _                     |
| Hindi Synthetic Error Corpus (ToolsForIL) | 20.5 K   | 77          | 15 K     | 75 K         | 16 K              | 5 K               | 10%                   |
| Hindi Pretrain Corpus                   | 370 K       | 77          | 40 K     | 5.5 M        | _                 | _                 | _                     |

Table 1: Dataset specification details
Table 2: Comparing BERT based variants. Typical word masking based pretraining has been used on all these variants. Real-Error (Fine Tuned) denotes fine tuning of the Bangla synthetic error dataset trained model on real error dataset, while Real-Error (No Fine Tune) means directly validating synthetic error dataset trained model on real error dataset without any further fine tuning.

| Spell Checker Architecture | Synthetic Error (Prothom-Alo) ACC | F1 | Real-Error (No Fine Tune) ACC | F1 | Real-Error (Fine Tuned) ACC | F1 | Synthetic Error (Hindi) ACC | F1 |
|---------------------------|---------------------------------|----|-------------------------------|----|---------------------------|----|-----------------------------|----|
| BERT Seq2seq              | 31.6%                           | 0.305 | 24.5%                        | 0.224 | 29.3%                     | 0.278 | 22.8%                       | 0.209 |
| BERT Base                 | 91.1%                           | 0.902 | 83%                          | 0.823 | 87.6%                     | 0.855 | 93.8%                       | 0.923 |
| Soft Masked BERT          | 92%                             | 0.919 | 84.2%                        | 0.832 | 88.1%                     | 0.862 | 94%                         | 0.933 |
| BSpell                    | 94.7%                           | 0.934 | 86.1%                        | 0.859 | 90.1%                     | 0.898 | 96.2%                       | 0.96  |

Table 3: Comparing BSpell exposed to various pretraining schemes

| Pretraining Scheme         | Synthetic Error (Prothom-Alo) ACC | F1 | Real-Error (No Fine Tune) ACC | F1 | Real-Error (Fine Tuned) ACC | F1 | Synthetic Error (Hindi) ACC | F1 |
|----------------------------|---------------------------------|----|-------------------------------|----|---------------------------|----|-----------------------------|----|
| Word Masking               | 94.7%                           | 0.934 | 86.1%                        | 0.859 | 90.1%                     | 0.898 | 96.2%                       | 0.96  |
| Character Masking          | 95.6%                           | 0.952 | 85.3%                        | 0.851 | 89.2%                     | 0.889 | 96.4%                       | 0.963 |
| Hybrid Masking             | 97.6%                           | 0.971 | 87.8%                        | 0.873 | 91.5%                     | 0.911 | 97.2%                       | 0.97  |

5.4 *BSpell vs Possible LSTM Variants*

BiLSTM is a many to many bidirectional LSTM (two layers) that takes in all *n* words of a sentence at once and predicts their correct version as output (Schuster and Paliwal, 1997). During SC, BiLSTM takes in both previous and post context into consideration besides the writing pattern of each word and shows reasonable performance (see Table 4). In Stacked BiLSTM, we stack twelve many to many bidirectional LSTMs instead of just two. We see marginal improvement in SC performance in spite of such large increase in parameter number. Attn_Seq2seq LSTM model utilizes attention mechanism at decoder side (Bahdanau et al., 2014). This model takes in misspelled sentence characters as input and provides the correct sequence of characters as output (Etoori et al., 2018). Due to word level spelling correction evaluation, this model faces the same problems as BERT Seq2seq model discussed in Subsection 5.2. Proposed BSpell outperforms these models by a large margin.

5.5 Ablation Study

BSpell has three unique features - (1) secondary branch with auxiliary loss (possible to remove this branch), (2) 1D CNN based SemanticNet sub-model (can be replaced by simple Byte Pair Encoding (BPE) (Vaswani et al., 2017)) and (3) hybrid pretraining (can be replaced by word masking based pretraining). Table 5 demonstrates the results we obtain after removing any one of these features. In all cases, the results show a downward trend compared to the original architecture.

5.6 Existing Bangla Spell Checkers vs BSpell

Phonetic rule based SC takes a Bangla phonetic rule based hard coded approach (Saha et al., 2019), where a hybrid of Soundex (UzZaman and Khan, 2004) and Metaphone (UzZaman and Khan, 2005) algorithm has been used. Clustering based SC on the other hand follows some predefined rules...
Table 4: Comparing LSTM based variants with hybrid pretrained BSpell. FastText word representation has been used with LSTM portion of each architecture.

| Spell Checker Architecture | Synthetic Error (Prothom-Alo) | Real-Error (No Fine Tune) | Real-Error (Fine Tuned) | Synthetic Error (Hindi) |
|----------------------------|--------------------------------|--------------------------|-------------------------|------------------------|
|                            | ACC   | F1    | ACC   | F1    | ACC   | F1    | ACC   | F1    |
| BiLSTM                     | 81.9% | 0.818 | 78.3% | 0.781 | 81.1% | 0.809 | 81.2% | 0.809 |
| Stacked BiLSTM             | 83.5% | 0.832 | 80.1% | 0.80  | 82.4% | 0.822 | 82.7% | 0.824 |
| Attn_Seq2seq (Char)        | 20.5% | 0.178 | 15.4% | 0.129 | 17.3% | 0.152 | 22.7% | 0.216 |
| BSpell                     | 97.6% | 0.971 | 87.8% | 0.873 | 91.5% | 0.911 | 97.2% | 0.97  |

Table 5: Comparing BSpell with its variants created by removing one of its novel features

| BSpell Variants | Synthetic Error (Prothom-Alo) | Real-Error (No Fine Tune) | Real-Error (Fine Tuned) | Synthetic Error (Hindi) |
|-----------------|--------------------------------|--------------------------|-------------------------|------------------------|
|                 | ACC   | F1    | ACC   | F1    | ACC   | F1    | ACC   | F1    |
| Original        | 97.6% | 0.971 | 87.8% | 0.873 | 91.5% | 0.911 | 97.2% | 0.97  |
| No Aux Loss     | 96.3% | 0.96  | 86.9% | 0.865 | 90.5% | 0.90  | 95.4% | 0.949 |
| No SemanticNet  | 94.5% | 0.94  | 85.7% | 0.848 | 89.2% | 0.885 | 95.2% | 0.95  |
| No Hybrid Pretrain | 94.7% | 0.934 | 86.1% | 0.859 | 90.1% | 0.898 | 96.2% | 0.96  |

Table 6: Existing Bangla spell checkers vs BSpell

| Spell Checker | Synthetic Error (Prothom-Alo) | Real-Error (No Fine Tune) |
|---------------|--------------------------------|--------------------------|
|               | ACC   | F1    | ACC   | F1    |
| Phonetic      | 61.2% | 0.582 | 43.5% | 0.401 |
| Clustering    | 52.3% | 0.501 | 44.2% | 0.412 |
| BSpell        | 97.6% | 0.971 | 87.8% | 0.873 |

on word cluster formation, distance measurement and correct word suggestion (Mandal and Hossain, 2017). Since these two SCs are not learning based, fine tuning is not applicable for them. They do not take misspelled word context into consideration while correcting that word. As a result, their performance is poor especially in Bangla real error dataset (see Table 6). BSpell outperforms these Bangla SCs by a wide margin.

5.7 Is BSpell Language Specific?

BSpell has originally been designed keeping the unique characteristics of Sanskrit originated languages such as Bangla and Hindi in mind. Here we see how this model performs on English which is very different from Bangla in terms of structure. We experiment on an English spelling error dataset published by Jayanthi et al. (2020). The training set consists of 1.6 million sentences. The authors created a confusion set consisting of 109K misspelled-correct word pairs for 17K popular English words. 20% of the words of the training set have been converted to spelling error based on this confusion set. The authors created BEA-60K test set from BEA-2019 shared task consisting of natural English spelling errors. The best correction rate achieved by the authors was around 80% using LSTM based ELMo model, whereas BSpell has achieved a correction rate of 86.2%. We have also experimented with BERT_Base model on this test set where we have used byte pair encoding as word representation. BERT_Base has achieved an error correction rate of 85.6%. It is clear that BSpell and BERT_Base do not have that much difference in performance when it comes to English compared to Bangla and Hindi.

5.8 Effectiveness of SemanticNet

The main motivation behind the inclusion of
SemanticNet in BSpell is to obtain vector representations of error words as close as possible to their corresponding correct words. We take 10 frequently occurring Bangla words and collect three real life error variations of each of these words. We produce SemanticVec representation of all 40 of these words using SemanticNet. We use principal component analysis (PCA) (Shlens, 2014) on each of these SemanticVecs and plot them in two dimensions. Finally, we implement K-Means Clustering algorithm using careful initialization with \( K = 10 \) (Chen and Xia, 2009). Figure 6 shows the 10 clusters obtained from this algorithm. Each cluster consists of a popular word and its three error variations. In all cases, the correct word and its three error versions are so close in the graph plot that they almost form a single point.

6 Conclusion

In this paper, we have proposed a SC named BSpell for Bangla and Hindi language. BSpell uses SemanticVec representation of input misspelled words and a specialized auxiliary loss for the enhancement of spelling correction performance. The model exploits the concept of hybrid masking based pretraining. We have also investigated into the limitations of existing Bangla SCs as well as other SOTA SCs proposed for high resource languages. BSpell has two main limitations - (a) it cannot handle accidental merge or split of words and (b) it cannot correct misspelled rare words. A potential research direction can be to eradicate these limitations by designing models that can perform prediction at sub-word level which includes white space characters and punctuation marks.

7 Limitations

BSpell model provides a word for word correction, i.e., number of input words and number of output words have to be exactly the same. Unfortunately, during accidental word merging or word splitting, number of input and output words differ and so in such cases BSpell will fail in resolving such errors. This type of error is more common in Chinese language. The advantage for us is that this type of error is rare in Bangla and Hindi as the words of these languages are clearly spaced in sentences. So, people will rarely perform accidental merge or split of words. Another limitation is that BSpell has been trained to correct only the top Bangla and Hindi words that cover 95% of the entire corpus.

As a result, this spell checker will face problems while correcting spelling errors in rare words. For such rare words, BSpell simply provides UNK as output which means that it is not sure what to do with these words. An advantage here is that most of these rare words are some form of proper nouns which should not be corrected and should ideally be left alone as they are. For example, someone may have an uncommon name. We do not want our model to correct that person’s name to some commonly used name.

An immediate research direction is to overcome the limitations of the proposed method. A straightforward way of dealing with the word merge, word split and rare word correction problem is to model spelling errors at character level (sequence-to-sequence type approach). We have taken this trivial attempt and have failed miserably (see the performance reported in the first row of Table 2). Solving these problems while maintaining the current spelling correction performance of BSpell can be a challenge. Another interesting future direction is to investigate on personalized Bangla and Hindi spell checker which has the ability to take user personal preference and writing behaviour into account. The main challenge here is to effectively utilize user provided data that must be collected in an online setting. Recently, deep learning based automatic grammatical error correction has gained a lot of attention in English language (Chollampatt and Ng, 2018), (Chollampatt and Ng, 2017), (Stahlberg and Kumar, 2021). SOTA grammar correction models developed for English can be trained and tested on Bangla and Hindi spell checking tasks as part of future research effort. Such benchmarking studies can play a vital role in pushing the boundaries of low resource language correction automation.

References

Ben Athiwaratkun, Andrew Gordon Wilson, and Anima Anandkumar. 2018. Probabilistic fasttext for multi-sense word embeddings. arXiv preprint arXiv:1806.02901.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. arXiv preprint arXiv:1607.06450.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

Zhang Chen and Shixiong Xia. 2009. K-means cluster-
Shamil Chollampatt and Hwee Tou Ng. 2017. Connecting the dots: Towards human-level grammatical error correction. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 327–333.

Shamil Chollampatt and Hwee Tou Ng. 2018. A multi-layer convolutional encoder-decoder neural network for grammatical error correction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Pravallika Etoori, Manoj Chinnakotla, and Radhika Mamidi. 2018. Automatic spelling correction for resource-scarce languages using deep learning. In *Proceedings of ACL 2018, Student Research Workshop*, pages 146–152, Melbourne, Australia. Association for Computational Linguistics.

Linyuan Gong, Di He, Zhuohan Li, Tao Qin, Liwei Wang, and Tieyan Liu. 2019. Efficient training of bert by progressively stacking. In *International Conference on Machine Learning*, pages 2337–2346. PMLR.

Yuzhong Hong, Xianguo Yu, Neng He, Nan Liu, and Junhui Liu. 2019. Faspell: A fast, adaptable, simple, powerful chinese spell checker based on dae-decoder paradigm. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 160–169.

Sadidul Islam, Mst Farhana Sarkar, Towhid Hussain, Md Mehedu Hasan, Dewan Md Farid, and Swakkhar Shatabda. 2018. Bangla sentence correction using deep neural network based sequence to sequence learning. In *21st International Conference of Computer and Information Technology (ICCCIT)*, pages 1–6. IEEE.

Sai Muralidhar Jayanthi, Danish Pruthi, and Graham Neubig. 2020. Neuspell: A neural spelling correction toolkit. *arXiv preprint arXiv:2010.11085*.

Nur Hossain Khan, Gonesh Chandra Saha, Bappa Sarker, and Md Habibur Rahman. 2014. Checking the correctness of bangla words using n-gram. *International Journal of Computer Application*, 89(11).

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Prianka Mandal and BM Mainul Hosssain. 2017. Clustering-based bangla spell checker. In *2017 IEEE International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*, pages 1–6. IEEE.

Jan Noyes. 1983. The qwerty keyboard: A review. *International Journal of Man-Machine Studies*, 18(3):265–281.

Sourav Saha, Faria Tabassum, Kowshik Saha, and Marjana Akter. 2019. BANGLA SPELL CHECKER AND SUGGESTION GENERATOR. Ph.D. thesis, United International University.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681.

Jonathon Shlens. 2014. A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*.

Md Habibur Rahman Sifat, Chowdhury Rafeed Rahman, Mohammad Rfasan, and Hasibur Rahman. 2020. Synthetic error dataset generation mimicking bengali writing pattern. In *2020 IEEE Region 10 Symposium (TENSYP)*, pages 1363–1366. IEEE.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Felix Stahlberg and Shankar Kumar. 2021. Synthetic data generation for grammatical error correction with tagged corruption models. *arXiv preprint arXiv:2105.13318*.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020. Ernie 2.0: A continual pre-training framework for language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8968–8975.

Naushad UzZaman and Mumit Khan. 2004. A bangla phonetic encoding for better spelling sugessions. Technical report, BRAC University.

Naushad UzZaman and Mumit Khan. 2005. A double metaphone encoding for approximate name searching and matching in bangla.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.
Dingmin Wang, Yi Tay, and Li Zhong. 2019. Confusionset-guided pointer networks for Chinese spelling check. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 5780–5785, Florence, Italy. Association for Computational Linguistics.

Jinhua Xiong, Qiao Zhang, Shuiyuan Zhang, Jianpeng Hou, and Xueqi Cheng. 2015. Hanspeller: a unified framework for chinese spelling correction. In International Journal of Computational Linguistics & Chinese Language Processing, Volume 20, Number 1, June 2015-Special Issue on Chinese as a Foreign Language.

Shaohua Zhang, Haoran Huang, Jicong Liu, and Hang Li. 2020. Spelling error correction with soft-masked bert. arXiv preprint arXiv:2005.07421.