Spell checker: Bidirectional Recurrent Neural Network based approach

Sonal Yadav
Computer Science
Lakehead University
1105500
syadav3@lakeheadu.ca

Anita Yadav
Computer Science
Lakehead University
1117437
ayadav3@lakeheadu.ca

Preetkumar Patel
Computer Science
Lakehead University
1107843
ppatel60@lakeheadu.ca

Sagar Deshpande
Computer Science
Lakehead University
1109819
sdeshpan@lakeheadu.ca

***Abstract*—A spell checker is a software tool which is designed to identify and correct any spelling errors in a text. Spell checkers can be paired with other programs, or can be independently distributed. Every spell checker uses a dictionary. In the dictionary each word from the text is looked up. It is recognized as error when a word is not in the dictionary. Basically the larger a spellchecker's dictionary is, the higher the rate of error detection; otherwise, misspellings would go undetected. A spell checker then scans the dictionary for the word that most matches the erroneous word, in order to correct the mistake. Moreover, every spell checker involves two main phases. The two main phases of the spell checker are error detection and error correction. In this research, we have implemented a spell checker which takes sentence with incorrect spelling as input and provides output as same sentence but the errors being corrected. The dataset used in the proposed method is taken from Gutenberg's project which is basically twenty popular books. The proposed method is based on Bidirectional Recurrent Neural Network(BRNN) which gives promising results.**

***Index Terms*—N-grams, Soundex, Convolution Neural Network, BRNN**

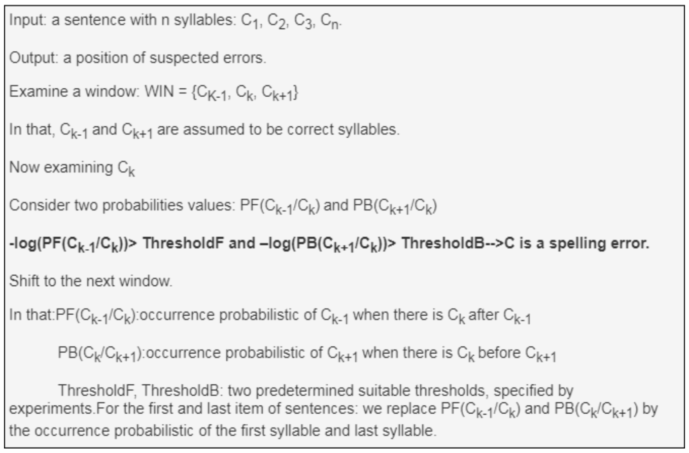
I. Introduction

Development of systems to improve human-computer interaction has always attracted researcher's attention in many areas of computer science. NLP (Natural Language Processing) systems are considered one of the most important research fields which reflect this interest. Basically, it is a type of human to computer interaction where the human language components, whether spoken or written, are formalized to allow a computer to complete value-adding tasks based on that interaction. The NLP group's goal is to design and build software that analyses, understands and generates languages that people use naturally, so that you can eventually address your computer as if you were addressing someone else. NLP has many applications including Automatic Summarization, Machine Translation, Retrieval of Data, Optical Recognition, and Question Answering. Spell-checking is yet another important application of computational linguistics, the work of which goes back to the early seventies when Ralph Gorin developed the first spell-checker for the DEC PDP-10 (Digital Equipment Corporation - Programmed Data Processor)mainframe computer at Stanford University [1]. By definition, a spell-checker is a computer program that identifies and often corrects wrong words in a text document [2]. Spelling errors can be categorized by word-creation: non-word errors and real word errors [3]. Non-word errors means that constructed words completely do not exist in the lexicon and other dictionaries. Basically, it is the misspelled words which occur due to typographical error. Real word errors are error that are made of words in the dictionary but are used in wrong context. If context is not considered then, we cannot decide whether or not they are spelling mistakes. With different types of spelling errors, we need various techniques that help to identify and correct the spelling errors which are made by the users. Furthermore, to correct these types of errors, every spell checker uses two main processes: error detection and error correction. These two phases of error detection and error correction is performed with the help of different components of spell checker. The spell checker consists of three main parts. The three main components of spell checker includes error detector, spelling generator and error corrector. Basically error detector detects the error or misspelled word and spelling generator gives suggestions about the correct word which should be used instead of the wrong word. Finally, error corrector chooses the correct or best word from the list generated by the spelling generator. All these three basic components are usually connected underneath an internal dictionary of words used to validate and look up spell-check words present in the text. However, as human languages are complex and contain countless words and terms, as well as domain-specific idioms, proper names, technical terminologies, and regular dictionaries are insufficient to cover all words in the vocabulary of the language [4]. In this research, we have developed a method to correct the sentences which consists of spelling errors. The proposed method uses twenty famous books for training the model from Gutenberg's project which is basically a repository of different type of books. The model uses BRNN network in the encoding layer and Bahdanau attention mechanism in the decoding layer. Basically, a sequence to sequence model is used as input is sequence of words and output is also sequence of words. Here, shorter sentences from the books are taken for training the model because it has fewer chances of including errors. Longer sentences are difficult to process because of its length and take more time during training the model. The methodology follows basic steps such as loading the data, preparing the data, building the model, training the Model, fixing custom sentences and summary. This step of proposed methodology is further discussed in the proposed methodology section. The proposed model is able to make correction of good quality and it is able to achieve good results.

II. Literature Review

A. Error Detection Phase

Prior to providing suggestions for spelling, spell correctors perform spell-checking. As the generation of suggestions for correct words is a computationally intensive process there is a need to detect the errors first [3]. The process of error detection typically checks if an input string is a valid index or the word is from dictionary. Spelling errors in typewritten text are typically in between 1% and 3% [5]. Most of the errors have one letter wrong which can be due to swapping of letters, addition of extra letters, or neglecting the letters [7]. Investigations show that out of all the spelling mistakes in the text being evaluated, 70% to 95% are single-error misspellings [5]. It was also observed that the 7.8% of spelling errors have their first alphabet wrong, whereas 11.7% of the errors had the second alphabet wrong [6]. There are several effective techniques available for identifying these types of errors. The most popular ones are –n-gram analysis and dictionary lookup. The text-recognition systems usually use n-gram techniques and spellcheckers utilize dictionary lookup method. In the research paper a syllable Bi-gram combining with a POS (Part of Speech) Bi-gram to detect spelling errors in the pre-processed document was used. The main advantage of N-gram algorithm is that it doesn't need word segmentation and can provide us high performance with a complete training corpus. Researcher's faced the challenge in getting a big training corpus and could not produce enough training data for Bi-gram model [8]. Therefore, the application of syllable Bi-gram was not as competent as anticipated. For improving the accuracy of the model POS data was added along with supplement words. 40 different POSs were identified in Vietnamese and used the POS labeled documents set. The co-occurrence frequencies of the POS were calculated in the POS Bi-gram window. These observed data were used for the verification of the doubtful error positions identified by the syllable Bi-gram as explained int the Fig 1 below [3]:



Input: a sentence with n syllables: C_1, C_2, C_3, C_n.

Output: a position of suspected errors.

Examine a window: WIN = {C_{K-1}, C_k, C_{k+1}}

In that, C_{k-1} and C_{k+1} are assumed to be correct syllables.

Now examining C_k

Consider two probabilities values: PF(C_{k-1}/C_k) and PB(C_{k+1}/C_k)

-log(PF(C_{k-1}/C_k))> ThresholdF and –log(PB(C_{k+1}/C_k))> ThresholdB-->C is a spelling error.

Shift to the next window.

In that:PF(C_{k-1}/C_k):occurrence probabilistic of C_{k-1} when there is C_k after C_{k-1}

PB(C_k/C_{k+1}):occurrence probabilistic of C_{k+1} when there is C_k before C_{k+1}

ThresholdF, ThresholdB: two predetermined suitable thresholds, specified by experiments.For the first and last item of sentences: we replace PF(C_{k-1}/C_k) and PB(C_k/C_{k+1}) by the occurrence probabilistic of the first syllable and last syllable.

Fig. 1. SYLLABLE BI-GRAM

The other technique of error detection – dictionary lookup – is based on the concept of finding words in the list of words which are assumed to be correct. The list here is called a dictionary and can be categorized in many ways. The various category of dictionary differs from each other on the basis of different characteristics like speed and storage requirements [8]. The words from specific fields like Economy or Computer Science can be combined with most commonly used words to form a large dictionary with huge number of correct words to be used as a reference. The drawback of these big dictionaries is that they require additional space and would need more time to search for the corresponding word. These dictionaries also come with the challenge of keeping them updated. They must be maintained to make sure the new terms for a particular field are considered and added to the dictionary so that it can cover the words in the text. Whereas, with too small dictionary user might get a lot of rejections/error detections for even the valid words. Therefore, the dictionary lookup must be tailored and should be constructed according to the purpose of the dictionary. In order to traverse these huge dictionaries and still maintain system response time, Hash Tables are the most commonly used technique to gain fast access to the dictionary. Hash address has to be computed for a string to be looked up in the dictionary and then the word stored at that address in the pre-constructed hash table is used for comparison. If the retrieved word is different from the evaluated string, a flag is raised for error detection. Hash tables with their random-access approach remove the burden of huge number of comparisons required for searching in the dictionary.

B. Error Correction Phase

Error Correction is the next step after error detection. There are a lot of different ways of approaching this problem. The early error correction algorithms don't take context into consideration when correcting the error. The process first generates a list of all the possible corrections, then it ranks these corrections. There are a lot of different methods for the generation and ranking of these corrections [8]. N-grams based approach is a common technique used for error correction. Here, N denotes the window size which will be placed on a sentence. The best value of N can be selected depending on the language or the training corpus uses a similar method, the model used is a 2-gram model which runs in parallel with the error detection uni-gram and a context sensitive error correction technique [1]. A variation of N-grams is using N-grams distance function to deal with non-word errors as explained by [3]. It is also important to note that for error correction, early algorithm like the Winnow-based spelling correction algorithm, still gives accurate results for English language. Another novel error correction technique is the minimum edit distance approach. In this approach there is an assumption that the amount of error made by a person is minimum. Hence the basic edit operation like insertion, deletion, substitution or transpose is used need to be minimum. The underlying assumption meaning that fewest errors have been made and hence the best solution is the one requiring the least combinations of the operations [8]. Bassil [1] makes use of minimum edit distance to get rid of outliers and reduce the number of possible corrections. This means that if there are data points which are furthest away from the minimum distance are not taken into consideration. A similar approach is used by Nguyen [3] but the weights associated for the operations (insertion, deletion,...) is not the same. They use weights which are based on the Vietnamese language and its grammar rules to calculate the minimum edit distance. Similar to minimum edit distance, the Levenshtein edit distance uses lexicon word to generate a similarity score. This score consists of operations (insertion, deletion,...), similar to minimum edit distance however, the weights associated for all the operations is the same the weights are all one [9]. The Agrep method is also similar to Edit distance, however for optimal performance the search criteria is different. An algorithm called Boyer-Moore algorithm is used to detect simple pattern errors [9]. SoundEx algorithm is used by Nguyen [3] combined with minimum edit distance to sort/ rank the word list. However, for these two methods to work together a heuristic function is applied. This heuristic function is based on Vietnamese language rules. Homonyms occur in the Vietnamese language similar to the English language. The accents and pronunciations for multiple regions is different than one another. This can cause tone errors or medial and principal vowel errors, which is why the SoundEx algorithm has been used. The utilization of neural networks for error correction is also a feasibility. Most techniques are based on back-propagation networks, usually utilizing one output for each word and one input for every N-gram for every position of that word. This method is different than the rest because it takes more time due to Neural Networks need to be trained before they can be used [8]. Hodge [9] have proposed using AURA (Advanced Uncertain Reasoning Architecture) for spelling modules, AURA is a collection of neural networks that could be implemented in modular way. The Correlation Matrix Memories (CMMs) with the help of supervised learning rule is used to map inputs and outputs. Correlation Matrix Memories is a part of AURA. They have used two CMMs, the first one is used for storing the words for N-grams and the hamming distance and the second one stores phonetic codes for homophone matching. The results from both of these combined will determine the score for the spell checking algorithm.

C. Available spell checkers

Spell checker is a well-known researched problem in Natural language processing. Most of the spell checking is done for the English language. but with the advent of more algorithms and techniques, researchers have come up with multilingual spell checkers. As different languages have different challenges, it is difficult to come up with a single solution. Following are listed some of the available spell checkers with a detailed comparative analysis between them.

1) JSpell: It is a spell-checking editor which accepts the input from the user and gives the suggestion of spellings when user clicks on the spell check button. It is a multilingual spell checker which detects English, French and German languages. Jspell allows adding frequent words in the dictionary and even corrects capitalization error [10].
2) Reverso Speller: It is a online tool which helps in online translation and spell checking for the text provided as input. It also provides with a online dictionary which helps in storing the word frequently used. It highlights the words with wrong spelling and on hovering gives suggestions. Along with the suggested correct word it also helps in translating to various other languages like French and provides added features of definition and synonym of the words.
3) SpellcheckPlus: This is another commonly used spell checker. It provides a user friendly and easy spell suggestion to the user. After a user inputs the text and clicks on spell check, the wrong words are highlighted with a yellow box and other words which can have multiple suggestions are highlighted with red. After the required correction, on clicking modify button, it replaces the wrong words with the suggested pop up words.
4) CheckDog: This tool works differently than the other tools, as it provides the spell check of the website. It just accepts the input as the website URL and then provides with the error in the websites if any. The other striking feature is that it helps you allow a custom dictionary as well to save for later use.
5) Bangla Spell checker: This spell checker mainly focuses on the Bangla language for the input. It serves both in online and offline mode. With the mis-spelt entered, it provides with the top 4 suggestions from the list of words. Like other online tools it also helps maintain a dictionary for the frequently used words [9].

III. Comparison Of Algorithms

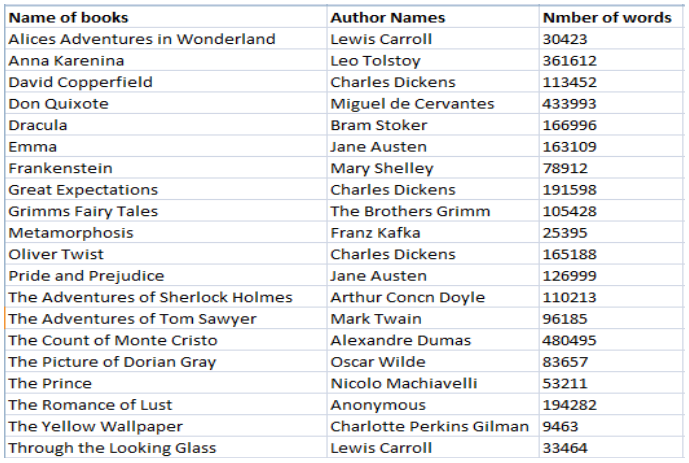
It is well known that each model will give different results for the different datasets. After hyper-tuning the parameters our model performs equally good if not better than others. However, the limiting factors are the complexity and the time taken. The parallel spell-checking algorithm was tested on 300,000-word articles and it was able to correct 94% of the total errors. The model they have used is a Bayesian-Noisy Channel model with the n-grams technique. This is modified with their own parallel detection and correction technique [1]. The Vietnamese spelling correction and detection technique used by Nguyen give 96% precision in English however it is significantly lower, this is because their method is more suitable for Asian languages more than English. They have used a bi-gram model with heuristic functions which calculates the weights using SoundEx and Minimum Edit Distance algorithms [3]. J. Duan model is a bidirectional LSTM with

CRF for Chinese language spelling checking. Their model is similar to our model, but they have tested their model on a Chinese language dataset instead of English. Their method gives them an accuracy of 88.6% for Sogou News dataset and 81.62% accuracy for literary Novel [12]. Our approach of using an RNN yields 92% accuracy but with hyper-tuning the parameters for this dataset we can get close to 94% accuracy.

IV. Proposed Methodology

A. Dataset Description

The dataset for training the model used for this project is taken from project Gutenberg. Basically, the data we have used is twenty popular books from project Gutenberg. The Name of books, author names and total number of words in each book used in the dataset is shown in the Figure. Here, text from all the books is converted to sentences. There are 132287 sentences used for training and testing the model.



Name of books	Author Names	Nmber of words
Alices Adventures in Wonderland	Lewis Carroll	30423
Anna Karenina	Leo Tolstoy	361612
David Copperfield	Charles Dickens	113452
Don Quixote	Miguel de Cervantes	433993
Dracula	Bram Stoker	166996
Emma	Jane Austen	163109
Frankenstein	Mary Shelley	78912
Great Expectations	Charles Dickens	191598
Grimms Fairy Tales	The Brothers Grimm	105428
Metamorphosis	Franz Kafka	25395
Oliver Twist	Charles Dickens	165188
Pride and Prejudice	Jane Austen	126999
The Adventures of Sherlock Holmes	Arthur Concn Doyle	110213
The Adventures of Tom Sawyer	Mark Twain	96185
The Count of Monte Cristo	Alexandre Dumas	480495
The Picture of Dorian Gray	Oscar Wilde	83657
The Prince	Nicolo Machiavelli	53211
The Romance of Lust	Anonymous	194282
The Yellow Wallpaper	Charlotte Perkins Gilman	9463
Through the Looking Glass	Lewis Carroll	33464

Fig. 2. Dataset Description

B. Implementation

The main aspect of any machine learning project is to have to good clean data. As in NLP projects it mainly deals with textual data, the need for error free datasets is a must. Hence in this project we introduce a spell checker which would alleviate all the problems of incorrect spellings in the texts. As discussed in the above sections where we introduced the types of spelling errors and also the various algorithms that can be used to correct them. Each algorithm has its own benefit with some drawbacks which make you choose the other algorithm. Amongst all the available spellcheckers discussed, we can see that spell checkers are not only famous for the common language like English but are also used for native languages. Though the performance and accuracy of word suggestion can be difficult for native languages, many steps are been taken in that direction to correct the same. Similarly, for English languages many approaches like a dictionary of learned words or else continuous learning with new data inputs is been formulated by the researchers. The model proposed is a BRNN based model which does pre-processing on the data and then passes the input to the model and gives a good accuracy for the corrected text. The reason to select BRNN for building the model is that it has lot of parameters which are easy to set up and can easily handle huge datasets. There is no need of prior data for spell correction and also can be easily extended to other tasks involving text summarization. In this model proposed we have used the existing built model and done parameter tuning on the layers and added more processing in the dataset and proposed a model for spelling correction [11].

The detailed steps in building the proposed model is as follows:

1) The import of all the necessary packages which will be required for building the model and also to read the data.
2) Then we make a folder for the dataset where a set of 20 books is loaded which will be used for training the model. The text of the files is loaded and saved in a list.
3) The next step involves cleaning of the data, so unwanted punctuation and spaces are removed from the dataset.
4) The data is not converted into lower case, because we would lose the essence of the data and would also miss out on many useful words.
5) A dictionary of integers is created which contains vocabulary that is converted to integers. Similarly, a separate dictionary of integers to vocabulary is also made.
6) Since the data consists of long texts, our system converts or splits the text from all the books into sentences.
7) The final input to the model is with integers hence after sentences creation we convert them to integers using the vocab_to_int functions.
8) The dataset is then split into train and test data for the training of the model and learning of the model for good accuracy.
9) Here since the dataset consists of all the grammatically corrects sentences, we introduce a method we add noises to thee words. These noises added are the general noises that can occur in any text document which has spelling errors. The noises that are added are swapping location of characters, adding additional characters and adding lower case letters to the existing words.
10) The next part is to build an BRNN model with all its parameters. As BRNN layer involves encoding of the input then processing and decoding the output. The methods for the same are created.
11) The last step is to create the layers of a sequence to sequence model using the above functions of encoder and decoder. Use of embedding and padding is done in the model building for fast processing and efficient results. The sentences are passed in batches to the model, with the additional noise that is added by the system.
12) The sequence to sequence model contains an encoding layer which is created using a Bi-directional BRNN which splits the neurons of the regular BRNN into two directions, one for positive time direction (forward

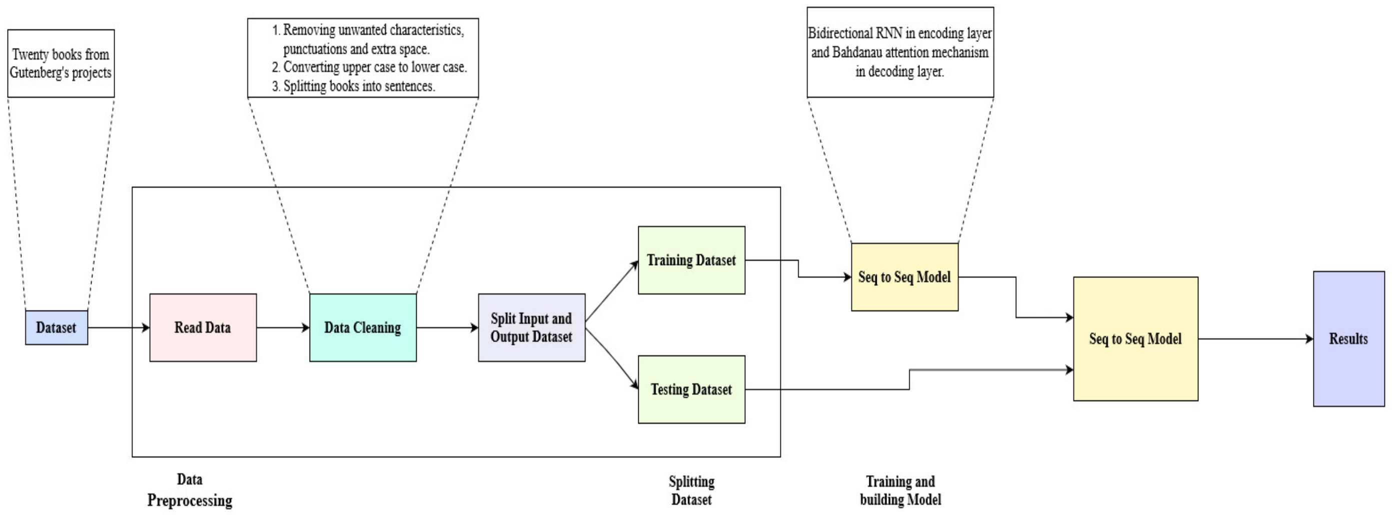

Twenty books from Gutenberg's projects
1. Removing unwanted characteristics, punctuations and extra space.
2. Converting upper case to lower case.
3. Splitting books into sentences.
Bidirectional RNN in encoding layer and Bahdanau attention mechanism in decoding layer.
Dataset
Read Data
Data Cleaning
Split Input and Output Dataset
Training Dataset
Testing Dataset
Seq to Seq Model
Seq to Seq Model
Results
Data Preprocessing
Splitting Dataset
Training and building Model

Fig. 3. Proposed Model

states) and another for negative time direction (backward states).

13) The decoding layer utilizes Bahdanau attention variant of the sequence to sequence model. This model uses the concatenation of the forward and backward sources hidden states in the bi-directional encoder and targets the hidden layers in their non-stacking single direction decoder.
14) The performance parameters are defined with low learning rate.
15) After the model is trained over a number of epochs, we can see that if the model learns something new in every epoch or not. Once the model has nothing new to learn it stops the training process and directly goes to the output cell.
16) For the output we can use two of the available options that is, user inputs a sentence of its own which is incorrect and our system outputs the correct sentence with all the spelling detected. The other is to use any of the available sentences from the dataset, add noise and then output the correct statement which is error free.

Since spelling errors, most of the time for spelling correctness checked in the context of surrounding words. Hence the more the model the trained the better the result we obtain. With a epoch of 100, the model gets to learn only a few features, so if the input sentences doesn't contain the word learned by the model, we might get absurd results which is not expected. Similarly, with higher epochs and more training parameters we can make our model more efficient to learn all the possible words available in the corpus and output an error free text.

V. Experimented Results

We use a collection of famous books as our dataset for testing the accuracy of our spell checker model. Pre-processing of the data was done to remove the unwanted words and also vocab to integer conversion was performed. Our data is tested on the model built using BRNN algorithm, with optimizing the hyper-parameters of the BRNN model. The output or the testing is displayed using the sentence corrected output and with the loss and accuracy metric. Our initial experiments displayed in Table 1 over the dataset analyse the representative power of BRNN. In all our results displayed, the baselines and our model learn over the training phase and get tuned with each epoch and are evaluated over the test phase. All the code is written TensorFlow, with different variations of BRNN filters, threshold value and strides in the layers. As displayed in the results we obtain the best performance by setting the threshold to value 99 and keeping 3 filters. We have also set our limit to input sentences to 150 words, which further enhances our performance. The Table 1 shows the comparative results of the baseline model and our improvement over the same. There are many parameters responsible for the performance of the BRNN model. Some of which we found majorly affected our results, were the threshold value and number of layers. Changing these parameters showed us a decrease in the loss and improvement in the accuracy along with better spelling correction for the input sentences.

TABLE I
Accuracy Metrics of the Model

Model	No.of Layers	Threshold	Accuracy
Baseline RNN	2	95	0.925
Proposed RNN	2	99	0.941
Proposed RNN	3	99	0.961

The accuracies in Table 2 shows the comparison of performance with the other available model. Some of the models discussed above are bayesian noisy model and some bi-gram model with heuristic functions and soundex for distance calculation. The table details the comparison based on the hyper-parameters like the filter size, threshold and BRNN sizes. From the comparative results we observe that the model proposed performs better with some parameter tuning and in some cases the other models give better accuracies.

TABLE II
COMPARISON OF PERFORMANCES WITH OTHER MODELS

Model	RNN sizes	Filters	Threshold	Accuracy
Bayesian Noisy Channel	-	5	-	0.94
Bi-Gram	-	5	-	0.89
Proposed RNN	-	-	99	0.961

We visualize that the training over the text dataset requires quite a bit of time. The encoder in the sequence to sequence BRNN model, requires learning all the feature of the data and we observe that the accuracy is based the word sequence. The training of the BRNN model involves number of runs base don the epoch mentioned by us. In every epoch the model trains itself to find new words and generate a loss and accuracy metric. This is done over few epochs and then when there is nothing new to learn, the training stops. The decoder layer of BRNN then comes into picture when the testing of any sentence from the corpus is performed. The output of any spell corrector is tested by the number of accurate corrections performed by the model. Thus based on all thee training the model outputs a very efficient result.

VI. Conclusion

The spell-checking algorithms are based on spelling errors generated in different languages. There are many spell checkers built in different languages such as Vietnamese, Persian, and Chinese. We have also listed the various spell checker available in different languages and the ways in which they detect the error and correct them. Our model takes a sentence which has a spelling error as input and outputs an error free sentence. The proposed model is an BRNN based model which after pre-processing is highly accurate. BRNN based models can perform exceptionally well on big datasets. Another benefit is that there is no need of prior data for spell correction and can be easily extended to other tasks involving text summarization. As the results show our model is similar in terms of accuracy when compared with other models and in some cases also gives better results.

References

[1] Bassil, Youssef. ”Parallel spell-checking algorithm based on Yahoo! Ngrams dataset.” arXiv preprint arXiv:1204.0184 (2012).

[2] Manning, Raghavan, Sch¨utze, “An Introduction to Information Retrieval”, Cambridge University Press, 2008

[3] Nguyen, Phuong Ngo, Thuan Phan, Dung Dinh, Thu Huynh, Thang. (2008)” Vietnamese spelling detection and correction using Bi-gram, Minimum Edit Distance, SoundEx algorithms with some additional heuristics “,96 - 102. 10.1109/RIVF.2008.4586339.

[4] M. S. Rasooli, O. Kahefi and B. Minaei-Bidgoli, ”Effect of adaptive spell checking in Persian,” 2011 7th International Conference on Natural Language Processing and Knowledge Engineering, Tokushima, 2011, pp. 161-164.

[5] Kukich, K., “Techniques for automatically correcting words in text”, ACM Computing Surveys, 24(4), 377–439, 1992.

[6] Joseph J. Pollock and Antonio Zamora, “Automatic spelling correction in scientific and scholarly text”, Commun. ACM, 27(4):358–368, 1984.

[7] Damerau, F.J., “A technique for computer detection and correction of spelling errors”, Comm. ACM 7(3):171-176, 1964.

[8] Gupta, Neha, and Pratistha Mathur. ”Spell checking techniques in NLP: a survey.” International Journal of Advanced Research in Computer Science and Software Engineering 2, no. 12 (2012).

[9] Hodge, Victoria J., and Jim Austin. ”A comparison of standard spell checking algorithms and a novel binary neural approach.” IEEE transactions on knowledge and data engineering 15, no. 5 (2003): 1073-1081.

[10] Shah, Khyati, Jikitsha Sheth, Mayuri Patel, and Kalpesh Lad. ”Comparative Study of Spell Checking Algorithms and Tools.” International Journal of Advanced Research in Computer Science 3, no. 3 (2012)

[11] Ghosh, Shaona, and Per Ola Kristensson. ”Neural networks for text correction and completion in keyboard decoding.” arXiv preprint arXiv:1709.06429 (2017)

[12] J. Duan, B. Wang, Z. Tan, X. Wei and H. Wang, ”Chinese Spelling Check via Bidirectional LSTM-CRF,” 2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC), Chongqing, China, 2019, pp. 1333-1336.

Appendix

A. Team Contributions

Our project consists of a thorough literature review, a baseline model identification, its implementation with tuning of hyperparameters of the model to obtain best performance followed by the report writing. Our team contribution for the tasks- Literature Review, Baseline Model Implementation, Tuning of Hyperparameter and Report writing were as follows:
Sonal Yadav- 25% Anita Yadav- 25%
Preetkumar Patel- 25% Sagar Deshpande- 25%

B. Github Link

https://github.com/ayadav3lake/NLP_Project_Group_9