Grammatical Error Correction in Sentences
Group 13

Aditya Vikram
14040

Ankit Kumar
14104

Borkar Pranay
14189

Ravish Kumar
14542

Rishabh Yadav
14554

Vivek Vishnoi
14818

1 Problem Statement

1.1 Motivation

In this digital era, where efforts are being made to digitize literature, we have to deal with large scale text data. It is important to develop automated tools for checking grammatical errors in long texts. Grammatical error detection is also useful in automation of error checks in scientific work like research papers, reports, etc. Grammatical error checkers in prominent text editors are outdated as they maintain hard-coded lists of errors, which are checked. With the advent of advanced NLP techniques and sequential learning, modeling grammatical errors has become increasingly efficient using Recurrent Neural Nets like *Long Short Term Memory (LSTM)* networks.

1.2 Background

We explored the previous work in the field of Grammatical Error Detection to get an idea of various types of errors. [1] describes in detail a wide range of errors present in English texts, and lists approximate percentages of each of these errors. We can get some intuition for these error types by taking a look at following examples:

It is obvious to see that **internet** saves people time and also connects people globally.
internet $\rightarrow$ *the internet*

This essay will **discuss about** whether a carrier should tell his relatives or not.
discuss about $\rightarrow$ *discuss*

These are examples of two prominent error types: articles and prepositions respectively. An intuitive approach for solving the problems is to use sequence based models to encapsulate the context of the words in order to check the grammatical mistakes.

1.3 Formal Statement

Given a text input, our aim is to predict the list of grammatical errors in sentences of the input. Explicitly, there are 28 types of grammatical errors present in our dataset. Among these type of errors, *Prepositional error* and *Article or determiner error* is our main focus, but we will also go for some other types of errors like, *verb form*, *noun number*, and *Subject Verb agreement*.

Our first aim is to identify the errors as precisely as possible, but if time permits we will also look forward to predict the possible corrections for the found error.

2 Related Work

2.1 Grammatical Error Identification as Sequence-to-Sequence Correction

[2] demonstrates an attention-based encoder-decoder model that can be used for sentence-level grammatical error identification. It also generates correction in addition to prediction of errors. They also found that character level model works better that word level model. They combined these two with a sentence level CNN model, and combining the three models gave the best results. Their results outperform other results on the AESW Shared Task on its own.

2.2 CoNLL-2013 - Shared Task on Grammatical Error Correction

[3] describes the CoNLL contest, where the goal is to automatically detect and correct grammatical errors present in English essays in NUS Corpus of Learner English (NUCLE). This deals with more error types, including noun number, verb form, and subject-verb agreement, besides much published research [4] on grammatical error correction focused on article and preposition errors. The approaches by various teams includes, building a classifier for each error type, Language modelling approach, etc.

2.3 The Automated Evaluation of Scientific Writing

[5] The task is aimed to promote development of evaluation tools in automated writings and assist authors to ensure their appropriateness in a scientific prose. It predicts sentences which require editing to ensure correctness and appropriateness, or needs improvement within scientific writing genre.

2.4 A Sentence Judgment System for Grammatical Error Detection

[6] This study used both rule-based and n-gram statistical methods to detect grammatical errors in Chinese sentences. The rule-based method identifies potential rule violations in sentences. The n-gram statistical method determines correctness of the input sentences using the n-gram scores of both correct and incorrect training sentences.

3 Proposed Algorithm

3.1 Grammatical Prediction

Based on the problem statement, our first task will be prediction of the types of grammatical errors that are present in the given sentence as input. The data-set that we are using is already annotated and it has a total of 28 types of grammatical error. Our main focus in this report will be based on detection of mainly 5 types of errors among those 28, because these errors are more often than the others.Those 5 errors are:-
Prepositional error, Article or determiner error,Verb form, Noun number, Subject Verb agreement.
To detect these 5 kinds of errors first we develop a combined model for all the error types which was trained on the complete corpus but because of high class imbalance present in the data-set more than one type of error present in the sentences. So we shift our strategy from training a combined model to trained 5 separate models for each one of the error type.

Each of these 5 models have following properties:-

1. Sentences have been converted into word vectors, using self-trained *Word2Vec* model from *gensim* library.
2. Stop words are included to retain context in order to perform sequence learning.
3. We've maintained the case of words because capital words have different context than small words (like they denote a name or the first word of a sentence).
4. Input sentences are zero-padded to get sentences with constant number of words (100).
5. Masking layer is added before *LSTM* layer to mask the padded zeros in the sequences.

Figure 1 illustrates the complete architecture of the model described above.

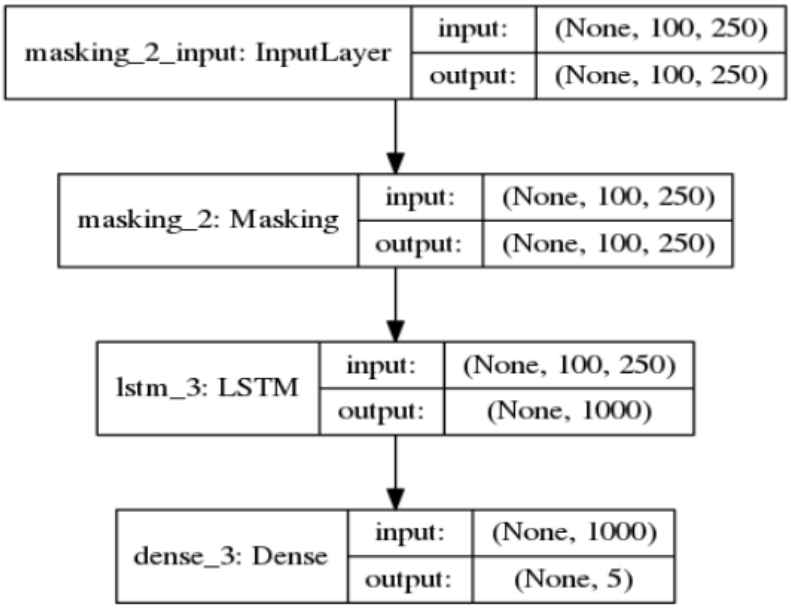
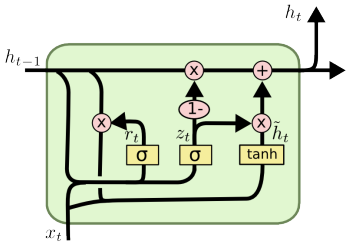


Figure 1: Keras prediction model plot

3.2 LSTM based Grammatical Correction

We follow the approach similar to that of proposed by Schmaltz et.al. wherein a sequence-to-sequence model (LSTM) is employed for encoding as well as decoding. Our aim would be to output a sequence of words in which grammatical error in that sentence is corrected.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figure 2: A sample LSTM model for correction

The LSTM encoder encodes the input sentence into a sequence of hidden layer vectors, associated with each word in the sentence, which are then decoded via a soft attention mechanism. The hidden states output by the encoder can be expressed as $h_i^s \in \mathcal{R}^n$ for each time step i, where

$$h_i^s = LSTM(h_{i-1}^s, x_i^s).$$

where $x_i^s \in \mathcal{R}^m$ is the word embedding for i^{th} word s_i and superscript s denotes the source. We set the initial state $h_0^s = 0$. $[h_1^s \cdots h_I^s]$ denotes the memory vector of this LSTM. The decoder LSTM produces a distribution over the next word (or tag) using the hidden states of encoder and the previously outputted words/tags related in same way as input:

$$h_j^t = LSTM(h_{j-1}^t, [x_j^t; c_{j-1}]).$$

where the hidden states of target h_i^t are combined with the memory vectors to produce context vectors c_j's. The final distribution over the next word is learnt by the model in similar manner.

4 Experiments

4.1 Dataset

We used *NUS Corpus of Learner English (NUCLE)* [7], the standard dataset for grammatical error correction. It was collected in a collaboration project between the NUS NLP Group led by Prof. Hwee Tou Ng and the NUS Centre for English Language Communication (CELC) led by Prof. Siew Mei Wu, as part of the PhD thesis research of Daniel Dahlmeier. NUCLE has the following skeleton:

- It consists of about 1400 essays written by learners of English on a wide range of topics, such as environmental pollution, health-care, etc.
- Mistakes are annotated with error tags from different categories of errors like Verbs, Subject-verb agreement, Articles/determiners, Nouns, etc.

4.2 Training

4.2.1 Grammatical Error Prediction

Each model was implemented using keras library and trained on a GPU Server. For each model, it took around 40 minutes to train per epoch with *batch_size = 32*. Number of epochs for individual model has been decided based on the divergence of the validation loss curve from train loss curve. It converged in about 5 epochs approximately. We tried different activation functions for the *Dense layer* as sigmoid failed in our case due to class-imbalance present in the data. Finally ReLU activation function has been used in the output layer.
To overcome the class imbalance in the data-set, we have used two approaches :-

Sub-sampling from correct sentences:-
Under this approach we have used correct and incorrect sentences in approximately 1:1 ratio. Training data contains 2k-5k incorrect sentences of any one error type, and similar number of incorrect sentences.

Using **class_weights** while training the model:-
Class weights has been used in the ratio 1:10 for correct and incorrect labels respectively. Training data contains 2k-5k incorrect sentences of any one error type, and 45k correct sentences.

4.2.2 Grammatical Error Correction

The grammatical correction in general is a difficult problem as the lengths of input and output sentences can be variable with no relation between them at all. We restrict experiment to a subset of problem in which the length of input sentence will be equal to the output sentence. So, firstly we extract all sentences that have corrected sentence of same length. Our entire model is implemented using the *keras* library. The two main components of seq2seq i.e. Encoder and Decoder are simultaneously trained. Input to the encoder is the list of word embedding vectors of words in incorrect sentence. The word embeddings are obtained by Word2Vec pre-trained model. Due to variable lengths of inputs, batch size is set to 1, i.e. each sentence is fed one by one. Input to the decoder is a list of one-hot vectors of words in correct sentence shifted by 1 time-step and prefixed by $$ (start token). Target output is simply the list of one-hot vectors of words in correct sentence.
For Example:

Correct sentence:	'He goes to school .'
Decoder input:	one hot(['$$', 'He', 'goes', 'to', 'school'])
Target:	one hot(['He', 'goes', 'to', 'school', '.'])

Training was done on 14k sentences. Among 14k there were about 7.2k incorrect sentences and remaining 7k correct sentences were sampled randomly. Training took about 3-4 sec per sentence per epoch on average. Training data was randomly shuffled. For 1 epoch training time was about 13 hrs. Training accuracy was seen to fluctuate initially as we fed only 1 sentence at each step. The variance in the training accuracy keeps on decreasing as we train model with more data. Towards end of training mean training accuracy was around 40%.

5 Results

5.1 Grammatical Error Prediction

Using Sub-sampling

	Accuracy	Precision
Training	68.1%	0.65
Testing	64.3%	0.59

Using class weights

	Accuracy	Precision
Training	78.0%	0.23
Testing	64.3%	0.59

Here, in the table we can see that the accuracy we are getting using class weights are relatively better than that of obtained using subsampling. But if we go by the precision, results of subsampling are are better. Overall we can prefer subsampling over using class weights.

5.2 Grammatical Correction

The results obtained were not quite satisfactory but some insights were obtained from the results. Some of the observations are as follows:

1. Every sentence starts with Capital letter. This can be because the previous word input is always same '$$' and is one hot vector.
2. First letter is predicted correctly in most of the given input sentences. This is because the input hidden vector that comes from encoder is fed as is to the first word.
3. The output sentences are similar in essence to that of the input sentence.
   For example:
   Input: Privacy should be compromise for security and safety
   Output: Surveillance should be banned to other problems
4. The stop words are repeated mostly and replace most of the words as their occurence is more as compared to other normal words.

The unsatisfactory results were also due to the scarcity of data as the data of 57000 sentence reduced to 14000 only due to the constraint of same length input and output sentences.

6 Possible Improvements

6.1 Grammatical Prediction

To improve the grammatical prediction part

1. A larger and balanced data-set can be used instead of the data-set which contains only few incorrect sentences.
2. Instead of accuracy metric, precision as a metric for cross-validation and testing can give better results.
3. If model can be trained on variable length inputs then performance can be significantly improved but at a cost of slow training.
4. Increasing model complexity by tuning model parameters.

6.2 Grammatical Correction

To improve the grammatical correction part the seq2seq model can be modified with the changes as proposed in [2].

1. The current target hidden state is linked with each of the hidden state corresponding to that word which was learned in the encoder. This is intuitive as in grammatical corrections, most of the words remain same or similar in the input and output sentence.
2. Each word is assigned a weight (which is learned) to avoid improper repetitions of the stop words. This is done as we can see a lot of repetitions of the stop words

7 Novelty

Instead of using the most probable decoding according to the sequence-to-sequence model, we are hoping to take advantage of the unique structure of the problem to impose the prior that all tokens in a decoded sequence should either exist in the input sequence or belong to a set of corrective tokens. The corrective token set is constructed during training and contains all tokens seen in the target, but not the source, for at least one sample in the training set. The intuition here is that the errors seen during training involve the misuse of a relatively small vocabulary of common words (e.g. the, an, their) and that the model should only be allowed to perform corrections in this domain. This prior is carried out through a modification to the decoding loop in sequence-to-sequence model.

8 Contribution

Aditya	20%
Ankit Kumar	20%
Pranay Borkar	20%
Ravish Raj	15%
Rishabh Yadav	5%
Vivek Vishnoi	20%

References

[1] Hwee Tou Ng et al. "The CoNLL-2014 shared task on grammatical error correction". In: *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*. 2014, pp. 1–14.

[2] Allen Schmaltz et al. "Sentence-level grammatical error identification as sequence-to-sequence correction". In: *arXiv preprint arXiv:1604.04677* (2016).

[3] Hwee Tou Ng et al. "The CoNLL-2013 Shared Task on Grammatical Error Correction". In: ().

[4] Daniel Dahlmeier and Hwee Tou Ng. "Grammatical error correction with alternating structure optimization". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics. 2011, pp. 915–923.

[5] Vidas Daudaravicius et al. "A report on the automatic evaluation of scientific writing shared task". In: *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*. 2016, pp. 53–62.

[6] Lung-Hao Lee et al. "A sentence judgment system for grammatical error detection". In: *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: System Demonstrations*. 2014, pp. 67–70.

[7] Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. "Building a large annotated corpus of learner english: The nus corpus of learner english". In: *Proceedings of the eighth workshop on innovative use of NLP for building educational applications*. 2013, pp. 22–31.