Sentence Auto Completion Using Stacked Bidirectional LSTM

Yash Srivastava, Nitin Malik
The NorthCap University, Gurugram, India

Abstract: Next word prediction systems are designed to autocomplete a sentence by predicting the next word or a phrase in a sentence. They are part of the applications such as search engines, messaging apps, and social media platforms. This paper makes use of a sequential language model such as bidirectional LSTM trained using backpropagation through time algorithm that will predict next words in a sentence. The study also investigates the impact of preprocessing techniques on the performance of the LSTM. The cross-entropy loss is used as an objective function. As a result, next word of a sentence is predicted based on the last three words of the sentence, accomplishing the goal of sentence auto completion. The results are compared with the existing literature and is found to be superior in performance.

Keywords: NLP, LSTM, Embedded layer, RNN, Next Word, Tokenizer.

1. Introduction

1.1 Motivation

The prediction systems allow for more efficient and accurate text input by predicting the most probable word to follow a given context. The basic principle is to detect the patterns and dependencies present in natural language to generate accurate next word prediction. Next word prediction systems have become an essential component of modern natural language processing (NLP) applications, enabling more efficient and accurate text analysis. Next word prediction systems have gained popularity in recent years, especially in the context of mobile devices and virtual assistants. These systems work by analysing the text input of the user and predicting the most likely next word or phrase based on the context [1] [2].

1.2 Literature Review

The paper [3] provides a method to predict characters which eventually lead it to form a word. The paper primarily focuses on sentiment analysis in English language texts and does not explore other languages. The paper [4] proposes a method for incorporating contextualized word embeddings in next word prediction model but does not compare the proposed approach with other existing approaches. The paper [5] introduces a model that combines recurrent neural networks (RNNs) with external memory for next word prediction but does not evaluate the proposed methodology on larger datasets.

The paper [6] makes use of RNN variant called gated recurrent units (GRUs) for next word prediction but does not compare the result with the existing literature. The paper [7] applies long short-term memory (LSTM) network in order to accomplish the desired task of next word prediction in spoken dialogue system but does not evaluate the proposed approach on larger datasets or compare it with other existing approaches. The paper [8] proposes next word prediction model that incorporates metadata information for predicting the next word.

The paper [9] proposes a method for next word prediction that uses convolutional neural networks (CNNs) and word embeddings. However, it does not compare the proposed approach with other existing approaches. The paper [10] investigates the use of character-level language models for next word prediction, it does not assess how well the suggested method works with regard to long-term dependencies.

The paper [11] introduces a method for incorporating semantic word embeddings in neural language models for next word prediction but does not evaluate the such as a gated recurrent unit (GRU), which was recently proposed, and a long short-term memory (LSTM) unit. proposed method on low-resource languages. The [12] Comparing various recurrent unit types in recurrent neural networks but does not mention any comparative analysis or evaluation of their proposed method with other existing methods.

1.3 Contribution

The main contribution are as follows:

a) LSTM overcomes the limitations of RNN.
b) Bidirectional LSTM model is mathematically modelled and is trained using Backpropagating through time (BPTT) algorithm
c) The cross-entropy loss obtained is lower than the published results in the existing literature.

2. Bidirectional LSTM Model

In RNN, since we are continuously writing to cell states, its continuously losing information it gained at previous timestep as the memory is limited. Moreover, there is vanishing gradient problem in RNN wherein gradient either vanishes or becomes very close to zero. This limitation is overcome in LSTM which ensures not all gradients vanishes.

LSTM [14] consists of three gates: Input Gate, Output Gate and Forget Gate. The input gate selectively reads the important bearing words. The output gate selectively writes in next cell state. The forget gate selectively forget stopwords and misleading words.

Bidirectional one more LSTM layer is added by LSTM, which changes the direction of information flow. which resulted in input sequence within the extra LSTM layer, flowing backward. Then the outputs from both LSTM layers is combined The Embedding layer looks for the embedding vector for each word-index using the integer-encoded vocabulary. [13].

The network is trained using backpropgation through time (BPTT) algorithm. The forward pass is computed at different timesteps followed by reverse pass in reverse timesteps. The gates regulate the information flow during forward pass and control the flow of gradients during reverse pass.

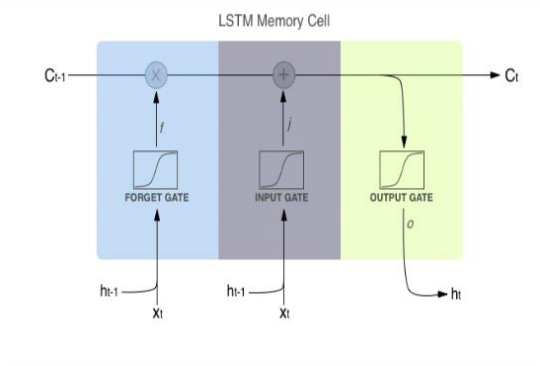

Fig.1: LSTM – Memory Cell

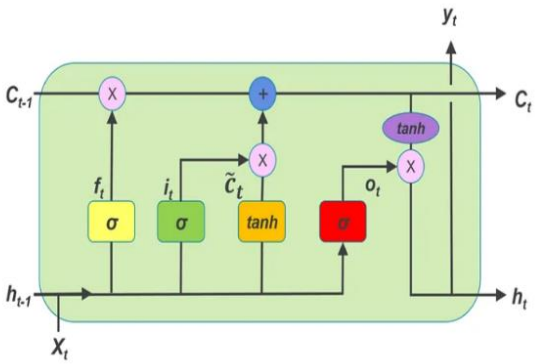

Fig.2: LSTM – Architecture

The input gate provides information regarding the fresh data that would be kept in the cell state, given by Eq. (1):

$$C_t' = \tan h\left(U_f X_t + h_{t-1} W_t + b_c\right)$$

Where The candidate for the cell at timestamp (t) is represented by C_t', the weight for the candidate is W_c is the weight for the candidate, h_{t-1} is the result of the LSTM block that came before it at timestamp (t-1), X_t is the input at current timestamp and b_c is the bias associated with that candidate.

The cell state c_t at timestamp t is given by Eq. (2):

$$C_t = f_t \odot C_{t-1} + i_t \odot C_t'$$

Where f_t represents the forget gate, C_{t-1}represents sate of cell at timestamp (t-1), i_t is the input gate and C_t' represents candidate at its timestamp's cell (t) and $\odot$ represents Hadamard product.

The formula for Forget Gate f_t is given by Eq .(3):

$$f_t = \sigma\left(X_t \odot U_f + h_{t-1} \odot W_f\right)$$

Where X_t is the current timestamp's input, U_f , is the input's weight, h_(t-1) is the previous timestamp's hidden state, W_t is the hidden state's weight matrix.

The Input Gate i_t is given by Eq. (4):

$$i_t = \sigma(X_t \odot U_i + h_{t-1} \odot W_t)$$

Where X_t is the input at timestamp t, U_i, is the weight matrix of the input, h_{t-1}, is the hidden state at timestamp t-1, Wi, Wi is the weight matrix of the input connected to the hidden state.

The output gate activates the LSTM block's final output at timestamp 't', which is provided by Eq. (5):

$$h_t = O_t \odot \tan h\,(C_t)$$

Where h_t, output of the current LSTM block at timestamp (t), O_t is the output gate and C_t represents state of cell at timestamp (t).

Loss is a measure of the error or discrepancy between a model's anticipated output and the actual output. The cross-entropy loss, which determines how closely the actual distribution of the following word differs from that anticipated by the prediction given the context, the cross-entropy H() between the Q and P probability distributions is given in Eq. (6):

$$H(P, Q) = -\sum P(x) \times \log(Q(x))$$

Where Q is an approximate representation of the target distribution and P could be the target distribution.

2.1 Dataset

```
The Project Gutenberg EBook of Index of the Project Gutenberg Works of
Arthur Conan Doyle, by Arthur Conan Doyle

This eBook is for the use of anyone anywhere in the United States and
most other parts of the world at no cost and with almost no restrictions
whatsoever.  You may copy it, give it away or re-use it under the terms
of the Project Gutenberg License included with this eBook or online at
www.gutenberg.org.  If you are not located in the United States, you'll
have to check the laws of the country where you are located before using
this ebook.


Title: Index of the Project Gutenberg Works of Arthur Conan Doyle

Author: Arthur Conan Doyle

Editor: David Widger

Release Date: December 30, 2018 [EBook #58574]

Language: English
```

Fig. 3: Dataset - E-book: The Adventures of Sherlock Holmes.

The Gutenberg Project's The Adventures of Sherlock Holmes by Arthur Conan Doyle is where the dataset [15] was created. The features of this dataset include the title, author, release date, and most recent update.

3. Solution Methodology

Step1: Import tensorflow library and keras API for preprocessing data, training and building model.

Step2: Store the text data in a list and convert it into a string. Preprocess the data such as stopword removal using regular expressions, reduce the words to its root word using lemmatization and remove the extra whitespaces and Unicode characters.

Step3: Tokenize the processed data using tokenizer and store it in a pickle file *token.pkl*. Vectorize a text corpus into a sequence of integers. The sequences are storred in array.

Step4: Design LSTM network as explained in section 2 using a sequence of embedded layer, two LSTM layer and two dense layers. Pass the vocabulary size and input length to the embedding layer.

Step5: Apply Adam optimizer as an optimizing function for compiling and subsequent learning of model at each iteration.

Step6: Train the model using BPTT algorithm as explained in Section 2. Save checkpoint at each epoch in a file *next_words.h5* and stores best fit model and evaluate weights at each iteration.

Step7: Evaluate the model performance using categorical cross entropy loss as an objective function

Step8: Load the model and tokenizer file for prediction using predict_next_words() function.

Step9: Iterate each item of dictionary that are in tokenizer file for subsequent prediction till you iterate over the whole dataset.

4. Result and Discussion

The length of the processed data is 573660. The length of the sequence is 108958. The vocabulary size is 8624. The LSTM layers has 1000 units. As indicated in Table 1, the first dense layer has 1000 neurons, while the second dense layer contains neurons of the same size as a vocabulary. Except for the output layer, which has softmax activation function, all the layers have ReLu activation functionality. A preset vocab size and a softmax activation function are both present in the output layer. ensures range of probability for each vocabulary size.

Layers	Output Shape	No. of trainable parameters
Embedding	(None, 3, 10)	86240
LSTM	(None, 3, 1000)	4044000
LSTM_1	(None, 3, 1000)	8004000
Dense	(None, 3, 1000)	1001000
Dense_1	(None, 3, 8624)	8632624
Total trainable parameters		21,767,864

Table 1: Model Structure

The rate of learning is 0.001. There are ten epochs and 64 is the batch size here. The output of the model is shown in Figure 4. The prediction is based on input sequence length of 3 words. The cross-entropy loss obtained at the end of the 10 epochs is 2.55 as given in Table 2. The graph is plotted in Table 3 and the result is compared with the existing results in Table 4.

```
Enter your line: This eBook is
['This', 'eBook', 'is']
1/1 [==============================] - 1s 661ms/step
for
Enter your line: The Adventure of the Engineer's
['of', 'the', 'Engineer's']
1/1 [==============================] - 0s 26ms/step
thumb
Enter your line: I had seen
['I', 'had', 'seen']
1/1 [==============================] - 0s 24ms/step
lord
Enter your line: I had seen
['I', 'had', 'seen']
1/1 [==============================] - 0s 27ms/step
lord
Enter your line: I have seldom
['I', 'have', 'seldom']
1/1 [==============================] - 0s 24ms/step
seen
Enter your line: 0
Execution completed.....
```

Fig. 4: LSTM model output

Epoch	Cross Entropy Loss
1	4.56
2	4.35
3	4.17
4	3.97
5	3.75
6	3.52
7	3.29
8	3.05
9	2.80
10	2.55

Table 2: Cross entropy loss at various epochs

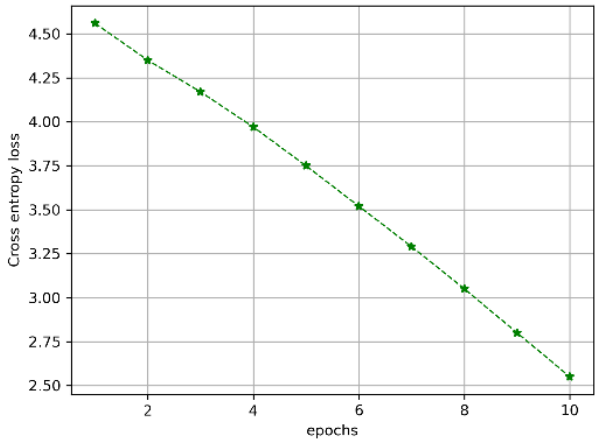

Table 3: Cross entropy loss vs epochs

Author	**Yash Srivastava & Nitin Malik. [proposed]**
Cross Entropy Loss	**2.55**

Table 4: Result Comparison

5. Conclusion

Next word prediction systems have become an essential part of modern NLP applications. These systems enable more efficient and accurate text input by predicting the most probable word to follow a given context. This paper makes use of the state-of-art language model bidirectional LSTM trained by BPTT algorithm to carry out prediction on the input sequence of given length. The model performance on the Cross-entropy loss outperforms the findings in the written research. The study also looked into how pre-processing methods affected performance. of the algorithm.

REFERENCES

[1] M. Soam and S. Thakur, "Next Word Prediction Using Deep Learning: A Comparative Study," *2022 12th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, Noida, India, 2022, pp. 653-658, doi: 10.1109/Confluence52989.2022.9734151.
https://ieeexplore.ieee.org/abstract/document/9734151
[2] Yash Patel, “Next Word Prediction BI-LSTM tutorial easy way”, 2021: https://www.kaggle.com/code/ysthehurricane/next-word-prediction-bi-lstm-tutorial-easy-way/notebook
[3] Keerthana N et al., “Next word prediction”, International Journal of Creative Research Thoughts, Vol 9, 2320-2882, 2021: https://ijcrt.org/papers/IJCRT2112562.pdf

[4] Vikas Raunak, Siddharth Dalmia, Vivek Gupta, and Florian Metze. 2020. On Long-Tailed Phenomena in Neural Machine Translation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3088–3095, Online. Association for Computational Linguistics.
https://www.aclweb.org/anthology/2020.findings-emnlp.276/
[5] Pengda Qin, Weiran Xu, and William Yang Wang. 2018. DSGAN: Generative Adversarial Training for Distant Supervision Relation_Extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 496–505, Melbourne, Australia. Association for Computational Linguistics.
https://www.aclweb.org/anthology/P18-1046/
[6] Oriol Vinyals et al., Sequence to Sequence Learning with Neural Networks, (2014) :
https://arxiv.org/abs/1409.3215
[7] Z. Shi, T. Nakano and M. Goto, "Instlistener: An Expressive Parameter Estimation System Imitating Human Performances of Monophonic Musical Instruments," *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Calgary, AB, Canada, 2018, pp. 581-585, doi: 10.1109/ICASSP.2018.8462141.
https://ieeexplore.ieee.org/document/8462141
[8] Chen Lin, Timothy Miller, Dmitriy Dligach, Steven Bethard, and Guergana Savova. 2016. Improving Temporal Relation Extraction with Training Instance Augmentation. In *Proceedings of the 15th Workshop on Biomedical Natural Language Processing*, pages 108–113, Berlin, Germany. Association for Computational Linguistics.
https://www.aclweb.org/anthology/W16-2914/
[9] Katharina Kann, Ryan Cotterell, and Hinrich Schütze. 2017. Neural Multi-Source Morphological Reinflection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 514–524, Valencia, Spain. Association for Computational Linguistics.
https://www.aclweb.org/anthology/E17-1049/
[10] Jaysidh Dumbali "Real Time Word Prediction Using N-Grams Model" International Journal of Innovative Technology and Exploring Engineering (IJITEE), Volume-8, Issue-5, (2019).
https://www.ijitee.org/wp-content/uploads/papers/v8i5/E3050038519.pdf
[11] ERHAN SEZERER et al., "Semantic Word Embeddings for Next-Word Prediction in Neural Language Models", 5 Oct (2021).
https://arxiv.org/pdf/2110.01804.pdf
[12] J. Chung et al., "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modelling", CoRR, Vol. Abs/1412.3555, (2014).
https://arxiv.org/abs/1412.3555
[13] Sawan Saxena, Understanding Embedding Layer, 2020
https://medium.com/analytics-vidhya/understanding-embedding-layer-in-keras-bbe3ff1327ce
[14] Diyanshu Thakur , LSTM, 2018
https://medium.com/@divyanshu132/lstm-and-its-equations-5ee9246d04af
[15] Dataset: E-Book, The Adventures of Sherlock Holmes by Sir Arthur Conan Doyle.
https://assets.datacamp.com/production/repositories/3937/datasets/213ca262bf6af12428d42842848464565f3d5504/sherlock.txt