Word Completion using LSTM

Muhammad Tayyab Sohail
Bachelors in Computer Science
FAST NUCES
Islamabad, Pakistan
I212478@nu.edu.pk

***Abstract*—This research develops a word-level Long Short-Term Memory (LSTM) model for sentence completion, trained on Shakespeare's plays. From a dataset of 111,396 rows, 11,000 were utilized for training the model. A user-friendly interface was created to provide real-time word suggestions as users type. The trained model achieves an impressive 90 percent accuracy, allowing users to input partial sentences and generating word predictions dynamically. Hyperparameter tuning was explored to optimize the model's performance, demonstrating strong coherence and fluency in sentence generation.**

I. Introduction

In this research, we developed a word-level Long Short-Term Memory (LSTM) model aimed at sentence completion. The model was trained on a text dataset of Shakespeare's plays, consisting of 111,396 rows, where we used 11,000 rows for model training. The task focused on predicting the next word in a sequence, with the goal of creating a user-friendly interface that dynamically suggests words as a user types.

The user interface updates in real-time, offering word suggestions based on the LSTM model's predictions. This feature allows users to input partial sentences, and the model autocompletes the sentence based on the previously trained data. Additionally, different hyperparameter configurations were explored to evaluate the model's coherence, fluency, and accuracy. The final model achieved an impressive 90 percent accuracy.

This paper elaborates on the model development, implementation, and user interface, as well as hyperparameter tuning and evaluation of sentence coherence.

II. Methodology

A. Dataset

The dataset used for training the LSTM model consists of Shakespeare's plays, comprising a total of 111,396 lines. Out of this, 11,000 rows were selected for training the model. The choice of dataset is significant, as it provides a wide range of vocabulary, style, and context, essential for developing a robust language model.

B. Preprocessing Steps

Preprocessing steps included:

- Text Cleaning: Removing punctuation, special characters, and converting text to lowercase.
- Tokenization: Splitting sentences into words to prepare for input into the LSTM model.
- Creating Sequences: Generating sequences of words to train the model on predicting the next word based on the previous context.

C. Model Architecture

The LSTM model consists of:

- An input layer for accepting the word sequences.
- One or more LSTM layers to capture the temporal dependencies of the sequences.
- A dense output layer with a softmax activation function for predicting the next word in the sequence.

III. LSTM Equations

The LSTM model's forward pass is governed by the following equations:

- Forget Gate:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

- Input Gate:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

- Cell State Update:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

- Output Gate:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

- Hidden State Update:

$$h_t = o_t * \tanh(C_t)$$

IV. Results

A. Training and Validation Loss and Accuracy

The model achieved a training accuracy of approximately 90 percent with a validation accuracy closely mirroring this value. Training loss decreased significantly over epochs, indicating effective learning.

V. Discussion

A. Analysis of Sentence Coherence

The predictions made by the LSTM model display a high degree of coherence, meaning the generated words align well with the expected linguistic structures.

B. Improvement Over Time

The model showed significant improvement in prediction accuracy over the course of training, indicating that with more data and training epochs, performance can continue to enhance.

C. Challenges Encountered

Challenges included:

- Balancing model complexity to avoid overfitting while maintaining a high level of accuracy.
- Ensuring that the model could generalize well across different contexts found in the training data.

VI. Conclusion

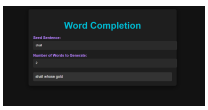
The development of a word-level LSTM model for sentence completion has demonstrated promising results. With an accuracy of 90 percent, the model effectively predicts the next word in a sentence based on context. The implementation of a user-friendly interface allows for real-time suggestions, improving user experience. Future work may involve fine-tuning the model further and exploring different architectures to enhance predictive performance.

Statistics

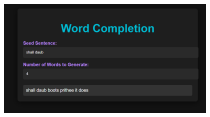
- **Total Lines:** 111,396
- **Total Words:** 3,661,570
- **Unique Words:** 27,381

VII. Prompts

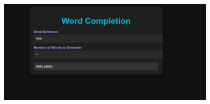
It asks the user to input the number of words to be generated as well as a seed sentence. These examples demonstrate the model's capability to generate contextually relevant words.



(a) 4 words to generate



(b) 2 words to generate



(c) 1 word to generate

VIII. Comparative Analysis of LSTM with Conventional RNN

Long Short-Term Memory (LSTM) networks improve upon conventional Recurrent Neural Networks (RNNs) by effectively addressing the vanishing gradient problem that RNNs encounter when learning long-term dependencies. While conventional RNNs maintain a simple structure where the hidden state is updated based on current input and previous hidden states, LSTMs utilize a more complex architecture with forget, input, and output gates. This gating mechanism enables LSTMs to selectively retain or discard information over extended sequences, making them better suited for tasks such as sentence completion.

In terms of performance, LSTMs demonstrate greater training stability and improved convergence compared to traditional RNNs, often resulting in lower training and validation losses. Although LSTMs are computationally more demanding due to their intricate architecture, the enhanced ability to capture context over longer sequences justifies the trade-off. Consequently, LSTMs have become the preferred choice for natural language processing applications, significantly advancing tasks that require understanding and predicting sequential data.

IX. References

- Ilaslan Düzgün, N. Next word prediction using LSTM with TensorFlow. Medium Article. https://medium.com/@ilaslanduzgun/next-word-prediction-using-lstm-with-tensorflow-e2a8f63b613c
- Kaggle. Shakespeare plays dataset. Dataset. https://www.kaggle.com/datasets/kingburrito666/shakespeare-plays
- Parmar, K. Sentence autocomplete using TensorFlow. Kaggle Code. https://www.kaggle.com/code/kparmar7/sentence-autocomplete-using-tensorflow