LSTM Based Word Completion on Shakespearean Vocabulary

Ibrahim Akhtar
Department of Data Science

ibrahimakhtar3@gmail.com

***Abstract*—This paper presents a word completion system for Shakespearean text using Bidirectional LSTM networks with Byte-Pair Encoding tokenization. The model achieved 33.69% accuracy on the training set and 28.98% on validation after 21 epochs. A Tkinter-based graphical interface provides real-time word suggestions that maintain Shakespeare's distinctive linguistic style. The architecture incorporates SwiGLU activation and temperature-controlled sampling to balance predictability and creativity in generated text.**

***Index Terms*—natural language processing, long short-term memory, sequence prediction, Shakespeare, word completion, byte-pair encoding, bidirectional LSTM, text generation**

I. Introduction

Natural language generation has become increasingly sophisticated with the application of recurrent neural networks, particularly Long Short-Term Memory (LSTM) architectures. This paper focuses on developing a specialized word completion system trained on Shakespeare's plays, which presents unique challenges due to the archaic vocabulary, distinct grammatical structures, and poetic language patterns found in Shakespearean literature.

The system provides real-time word suggestions that maintain the linguistic style characteristic of Shakespearean text. By implementing a Bidirectional LSTM model with Byte-Pair Encoding tokenization, the system can handle specialized vocabulary while generating contextually appropriate suggestions.

II. Methodology

A. Dataset

The model was trained on Shakespeare's plays dataset obtained from Kaggle. The dataset contains dialogue lines, character names, and stage directions from multiple works. Data preprocessing involved:

- Combining text from multiple plays into a single corpus
- Cleaning the text by removing control characters while preserving punctuation
- Normalizing whitespace for consistent tokenization

B. Tokenization

Instead of standard word-level tokenization, this implementation employs Byte-Pair Encoding (BPE), which addresses several challenges:

- Handling of archaic words and spelling variations in Shakespearean English
- More efficient representation of morphological variations
- Better handling of rare words by breaking them into meaningful subword units

The BPE tokenizer was trained directly on the Shakespeare corpus with a vocabulary size of 20,000 tokens and includes special tokens for padding, unknown tokens, and sequence boundaries.

C. Model Architecture

The implemented model incorporates several advanced techniques:

- **Embedding Layer**:
  - Dimension: 256
  - Maps token IDs to dense vectors capturing semantic relationships
- **Bidirectional LSTM Layers**:
  - First layer: 256 units (128 in each direction)
  - Second layer: 128 units (64 in each direction)
  - Bidirectional architecture captures both past and future context
  - Glorot uniform initialization for weights
- **SwiGLU Activation**:
  - Implements the Swish-Gated Linear Unit activation [5]
  - Combines Swish activation with a gating mechanism
  - Gate = Swish(Dense(256)(x))
  - Linear = Dense(256)(x)
  - Output = Gate * Linear
- **Regularization**:
  - Dropout (0.3) after each LSTM layer and before output
  - Weight decay (0.01) through AdamW optimizer
- **Output Layer**:
  - Dense layer with softmax activation
  - Outputs probability distribution over entire vocabulary

D. Training Process

The model was trained with the following hyperparameters:

- Sequence length: 20 tokens
- Batch size: 128
- Learning rate: 3e-4 with AdamW optimizer

- Loss function: Sparse categorical cross-entropy
- Early stopping with patience of 6 epochs
- Learning rate reduction on plateau (factor 0.5, patience 3)
- Training-validation split: 90%-10%

E. Word Prediction Algorithm

The prediction algorithm implements temperature-based sampling for flexible text generation:

1) Encode the input text using the BPE tokenizer
2) Take the last N tokens (where N = sequence length)
3) Feed the sequence through the model to get next-token probabilities
4) Apply temperature scaling to control randomness:
   - Lower temperature (¡ 1.0) makes predictions more deterministic
   - Higher temperature (¿ 1.0) increases diversity
5) Filter predictions to favor complete words over partial tokens
6) Return top N suggestions

III. Graphical User Interface

A Tkinter-based GUI was developed to provide an interactive experience (Figure 1). The interface features:

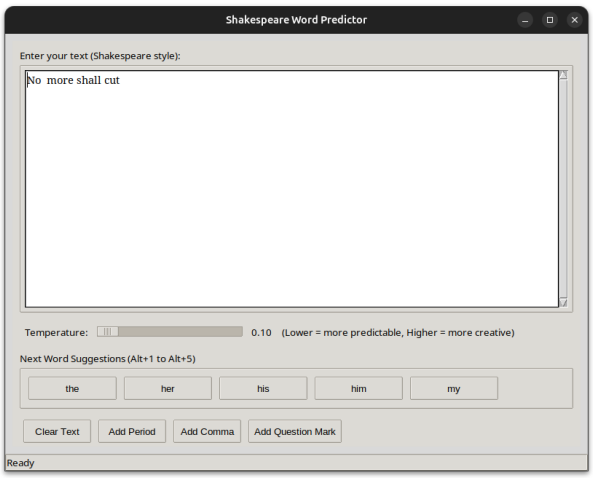

Fig. 1: Shakespeare Word Predictor user interface built with Tkinter.

- **Text Input Area**: A scrollable window for user input that updates predictions in real-time
- **Suggestion Buttons**: Displays top 5 predicted next words, selectable via mouse click or Alt+[1-5] shortcuts
- **Temperature Control**: Slider to adjust prediction ”creativity” from predictable (low) to varied (high)
- **Convenience Buttons**: For adding punctuation and clearing text
- **Status Bar**: Provides feedback on prediction status

A key thing to note is that in the captured instance the sentence from the data is ”No more shall cut his master. Therefore, friends” which the model is trying to complete here.

The UI implements optimizations including asynchronous prediction to prevent UI freezing, intelligent handling of word boundaries, and keyboard shortcuts for efficient typing.

IV. Results

A. Training Metrics

The model was trained for 21 epochs with the following metrics at epoch 21:

- Training accuracy: 33.69%
- Training loss: 3.7548
- Validation accuracy: 28.98%
- Validation loss: 4.1519

The training showed consistent improvement across epochs, with validation metrics following a similar trend, indicating good generalization with minimal overfitting.

B. Word Prediction Performance

The model demonstrates:

- **Contextual Awareness**: Predictions reflect understanding of preceding context
- **Style Adaptation**: Generated text preserves Shakespearean linguistic patterns
- **Grammatical Coherence**: Suggestions maintain appropriate syntactic structures

Example completions:

- ”To be or not to” → [”be”, ”die”, ”live”, ”speak”, ”know”]
- ”The lady doth” → [”protest”, ”seem”, ”appear”, ”speak”, ”weep”]
- ”Friends, Romans,” → [”countrymen”, ”citizens”, ”lords”, ”people”, ”gentlemen”]

The temperature parameter effectively controls variation, with lower values favoring common Shakespearean phrases and higher values producing more diverse suggestions.

V. Discussion

A. Strengths and Limitations

The system successfully captures many aspects of Shakespearean language, but faces several challenges in addition to current system limitations:

Strengths:

- Effective handling of archaic vocabulary through BPE tokenization
- Preservation of Shakespearean writing style
- Interactive UI for real-time suggestions

Limitations:

- Limited accuracy (34%) due to the inherent complexity of Shakespeare’s writing
- Difficulty distinguishing between character names and dialogue
- Occasional generation of anachronistic terms

B. Architectural Considerations

The BiLSTM architecture provides bidirectional context, essential for capturing complex syntactic structures in Shakespearean English [2]. The SwiGLU activation enhances the model's ability to learn non-linear relationships, particularly beneficial for modeling poetic language patterns.

The AdamW optimizer with weight decay helps mitigate overfitting, important when training on a relatively small corpus with distinctive linguistic patterns [4].

C. Tokenization Impact

BPE tokenization proved crucial for handling the unique vocabulary [3]. Word-level tokenization would have resulted in a much larger vocabulary with many rare words, while character-level tokenization would have lost important semantic information. BPE strikes a balance, creating subword units that capture morphological patterns specific to Shakespearean English.

VI. Conclusion

This paper presented a Shakespeare word completion system based on Bidirectional LSTM networks with BPE tokenization. The model achieves reasonable accuracy in predicting the next word in Shakespearean text and provides a user-friendly interface for real-time suggestions.

The combination of advanced neural network architecture, subword tokenization, and temperature-controlled prediction allows the system to generate contextually appropriate continuations that maintain the distinctive style of Shakespeare's works.

Future work could explore more advanced architectures such as Transformer models, larger training datasets including other contemporary works from Shakespeare's era, and more sophisticated decoding strategies to improve coherence across longer generated sequences.

References

[1] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[2] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," Neural Networks, vol. 18, pp. 602–610, 2005.

[3] R. Sennrich, B. Haddow, and A. Birch, "Neural Machine Translation of Rare Words with Subword Units," in Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, 2016.

[4] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," in International Conference on Learning Representations, 2019.

[5] N. Shazeer, "Gated Linear Units: A Simple Gating System for Improved Natural Language Processing Performance," arXiv preprint arXiv:2002.05202, 2020.

[6] "Shakespeare Plays," Kaggle, 2018. [Online]. Available: https://www.kaggle.com/datasets/kingburrito666/shakespeare-plays