Project Title: Next Token Prediction in Classic Literature

Objective:

To compare the effectiveness of word-level vs. subword-level (BPE) tokenization in predicting the next token using an LSTM-based language model.

Dataset:

- Source: Kaggle
- File: Sherlock Holmes.txt
- Size: 610.92 kB
- Language: Classic 19th-century English
- Preprocessed Word Count: ~536,000 words
- Unique Words: 7,901
- Total Sentences: 7,278

Data Preprocessing:

- Removed special characters, digits, and excessive whitespace
- Lowercased all text
- About 1.5% of words were removed
- Tokenization performed at:
  - Word-level (using Keras Tokenizer)
  - Subword-level (using SentencePiece BPE with 8000 merge operations)

Model Architecture:

- Model Type: Bidirectional LSTM
- Embedding Dimension: 100
- Hidden Units: 128

- Dropout: 0.2
- Loss Function: Categorical Crossentropy
- Optimizer: Adam
- Epochs: 20

Training Setup:

Both models were trained on input-output pairs formed using a sliding window over the token sequences, predicting the next token given the previous ones.

Results:

Tokenization Method	Initial Loss	Final Loss	Initial Accuracy	Final Accuracy
Word-level	6.6	0.34	0.06	0.95
Subword (BPE)	6.6	3.00	0.05	0.52

Observations:

- Word-level tokenization gave significantly better performance.
- BPE helps with rare word generalization but has lower overall performance.

Conclusion:

This project showed how the way we break down text (tokenization) can really affect how well a model predicts the next word. The word-level method gave the best results because it works directly with full words, which fits well with the style of the Sherlock Holmes text. The BPE method didn't perform as well, but it's still useful for handling rare or unknown words.

Used Tools:

- Python

- Google Colab

- Keras (for LSTM and Tokenizer)

- SentencePiece (for BPE tokenization)

- Matplotlib (for plotting training results)

- NumPy & Pandas (for data handling)