Jaime Song

Assignment 4 Technical Report

Dataset Overview:

For the sequence modeling portion of this assignment, I selected William Shakespeare's *Romeo and Juliet* from Project Gutenberg as the text corpus. The dataset consists entirely of unstructured text. Unlike typical tabular datasets, this dataset contains a single variable.

Analysis:

I began by inspecting the raw text and removing the Project Gutenberg license headers, metadata, and ending markers. After cleaning, I performed the following steps.

- Converted all text to lowercase for consistency
- Tokenized the text into individual words
- Constructed a vocabulary of unique words
- Mapped each word to a numeric index

The final vocabulary contained several thousand unique word tokens. A count of the most common words showed the expected prominence of character names and dialogue markers.

Sequence Preparation

To train a word level prediction model, I divided the text into overlapping input-output pairs. Each input sequence consisted of 30 consecutive words, and the corresponding output was the next word in the text. This sequence length is one of my three required changes from the reference notebook.

Methodology

Neural Network Architecture (LSTM Model)

I implemented a recurrent neural network using Keras, following the general structure of the reference code but applying three modifications as required:

Modification 1: New Dataset

- I replaced the reference dataset with Romeo and Juliet

Modification 2: Increased Sequence Length

- The reference notebook used a sequence length of 20. I increased it to 30 words

- This gives the model more contextual history, which often helps with long term structure and improved coherence in generated text

Modification 3: Additional LSTM Layer

- Added a second LSTM layer with return_sequences = TRUE, creating a deeper network that can model more abstract linguistic patterns

<u>Transformer Models</u>

For the second portion of the assignment, I selected three pre-trained Transformer models from HuggingFace and demonstrated one usage example for each:

Model 1: DistilBERT

- Type: Encoder only Transformer
- Original training: Masked language modeling and next sentence prediction on large English corpora
- Example task: Sentiment classification on a short movie review

Model 2: GPT-2

- Type: Decoder only Transformer
- Original training: Predict the next token over 40 GB of internet text
- Example task: Free form text generation from a short prompt

Model 3: BART

- Type: Encoder decoder Transformer
- Original training: Denoising autoencoding and sequence to sequence tasks

Each model was loaded using the HuggingFace pipeline() interface for simplicity. Inputs and outputs were clearly displayed in the demonstration notebook.

Results

<u>LSTM Word Prediction Results</u>

After training, the LSTM model successfully generated sequences of Shakespeare-like text. Although the output is not grammatically perfect, it captures elements such as dramatic diction, repetition patterns, and frequent references to common characters.

Changing the sequence length and adding a second LSTM layer increased training time but generally improved coherence. The model occasionally fixated on specific words due to vocabulary imbalance, which is expected in small literary corpora.

Transformer Model Results

Each pretrained Transformer produced strong results without additional training:

- DistilBERT correctly identified sentiment in test sentences
- GBT-2 generated plausible creative text from prompts
- BART produced concise summaries that captured the main ideas of longer inputs

These results highlight the advantage of large scale pre-training compared to training an LSTM from scratch.

Reflection

This assignment gave me hands-on experience with two very different approaches to natural language processing. Building the LSTM model taught me how much preprocessing matters in sequence tasks and how changes in architecture (such as increasing sequence length or adding layers) influence the model's ability to learn context.

I also realized that even small errors in preprocessing can break the model or significantly change the results. Debugging the text generator especially taught me how fragile sequence-to-sequence pipelines can be.

Working with Transformers highlighted how much modern NLP relies on large-scale pretraining. The difference in performance between my custom LSTM and the HuggingFace models is dramatic. If I were to approach a similar problem in the future, I would likely begin with a Transformer architecture rather than training a model entirely from scratch.

Overall, this project strengthened my understanding of sequential models, tokenization, and the practical challenges of building NLP systems end-to-end.