Ngoc Thanh Uyen Ho
A1875049

Assignment 3 Question 5

Language Modelling with RNN (LSTM) and Transformer

I. Objective

The aim of this task was to train and compare two small language models one based on an LSTM and one using a Transformer on the same dataset and tokenization setup. Both models were trained on the Shakespeare corpus (input.txt), which was tokenized using Byte Pair Encoding (BPE) with a vocabulary size of 10,000. The tokenizer was implemented with SentencePiece to make sure both models used exactly the same subword representation. The goal was to observe differences in training behaviour, efficiency, and final validation performance.

II. Implementation Summary

For tokenization, I first trained a 10k BPE model on the dataset. This produced a file bpe_tokens.pt with around 274k tokens, and both models used these tokens for training.

LSTM model:

- One LSTM layer (hidden size = 256, embedding size = 128)
- Optimized with AdamW (learning rate = 3e-4)
- Processes each token sequentially and keeps a hidden state through time

Transformer model:

- Two Transformer blocks with 4-head self-attention and feed-forward layers
- Embedding size = 128, block size = 64
- Also optimized with AdamW (3e-4)
- Learns relationships between tokens in parallel rather than step by step

Both models were trained for about 1,000 steps on CPU, with a batch size of 16 and a sequence length of 64. Evaluation was done every 200 steps using validation loss and perplexity.

III. Results

Model	Training Behaviour	Final Val. Loss	Perplexity	Time
LSTM	Loss dropped from 9.22 → 6.10	6.0980	444.97	185.7s
Transformer	Loss dropped from 9.32 → 5.91	5.9063	367.36	148.5s

Ngoc Thanh Uyen Ho
A1875049

Example logs:

LSTM step 1000 | train 5.9233 | val 6.0980 | ppl 444.97
Transformer step 1000 | train 5.5521 | val 5.9063 | ppl 367.36

Both models trained smoothly without instability. The Transformer reached slightly lower loss and perplexity, meaning it captured context a bit better overall.

IV. Analysis

The LSTM trained slower since it processes tokens one by one, while the Transformer could handle them in parallel, which made it faster even on CPU. The Transformer also achieved better perplexity (367 vs 445), suggesting it learned the structure of the text more effectively.

However, on this small dataset, the difference wasn't huge. The LSTM still performed quite well and was easier to train. Transformers usually show bigger advantages on large datasets or longer sequences because they can attend to the whole context, while LSTMs can only remember information through hidden states.

In terms of computation, the LSTM was lighter, while the Transformer needed more operations per step due to self-attention's $O(n^2)$ complexity. But if trained on GPU, the Transformer would scale much better because of its parallelism.

V. Conclusion

Both models learned the language patterns in the Shakespeare dataset successfully. The Transformer achieved lower validation loss and perplexity, showing stronger ability to model long-range context, while the LSTM was faster to converge and required fewer resources.

Overall, these results match what is seen in NLP research today: LSTMs are still efficient and simple for small tasks, but Transformers perform better when modelling complex or large-scale data because of their attention mechanism and parallel computation.