CSC 4700 Foundational AI Project 2 Report

Catherine Rodriquez
CSC 4700
Due: 4/16/2025

Abstract

In this project, I implemented and trained three small-scale language models based on classic neural network architectures: Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), and Transformer. The models were built using PyTorch and trained on a text dataset compiled from the Project Gutenberg online library. This gave me valuable hands-on experience with core model structures, training workflows, and evaluation techniques. Performance was measured using Perplexity (PPL) and BLEU scores, where lower PPL and higher BLEU indicate better text generation. The Transformer model performed the best, achieving a PPL of 101.24 and a BLEU score of 0.0119. A full comparison of PPL and BLEU scores across all models is provided in the Results section below.

Methodology

I began this project by implementing the Recurrent Neural Network (RNN), as I believed it would be the most approachable model to start with while building out the core functionality of the system. I recalled Dr. Ghawaly's suggestion from class to use a class-based structure to reduce redundancy and improve scalability. After completing the initial version of the RNN, I reorganized the project using object-oriented design principles. This was my first time applying these concepts to a project of this size, so I started by outlining the key components and identifying shared functionality that could be reused across models. This planning led to the creation of a base_model module to define common methods and interfaces, as well as separate modules for each model type: rnn.py, lstm.py, and transformer.py. I also created a language_dataset module and a tokenizer to manage data preprocessing and ensure consistency across models.

Alongside model development, I wrote three utility scripts located in the root directory: train.py, eval_models.py, and generate_text.py. The train.py script allows users to train a selected model using the dataset. It loads the tokenizer and dataset, initializes the chosen model, and manages the training loop, including logging and saving checkpoints. The eval_models.py script

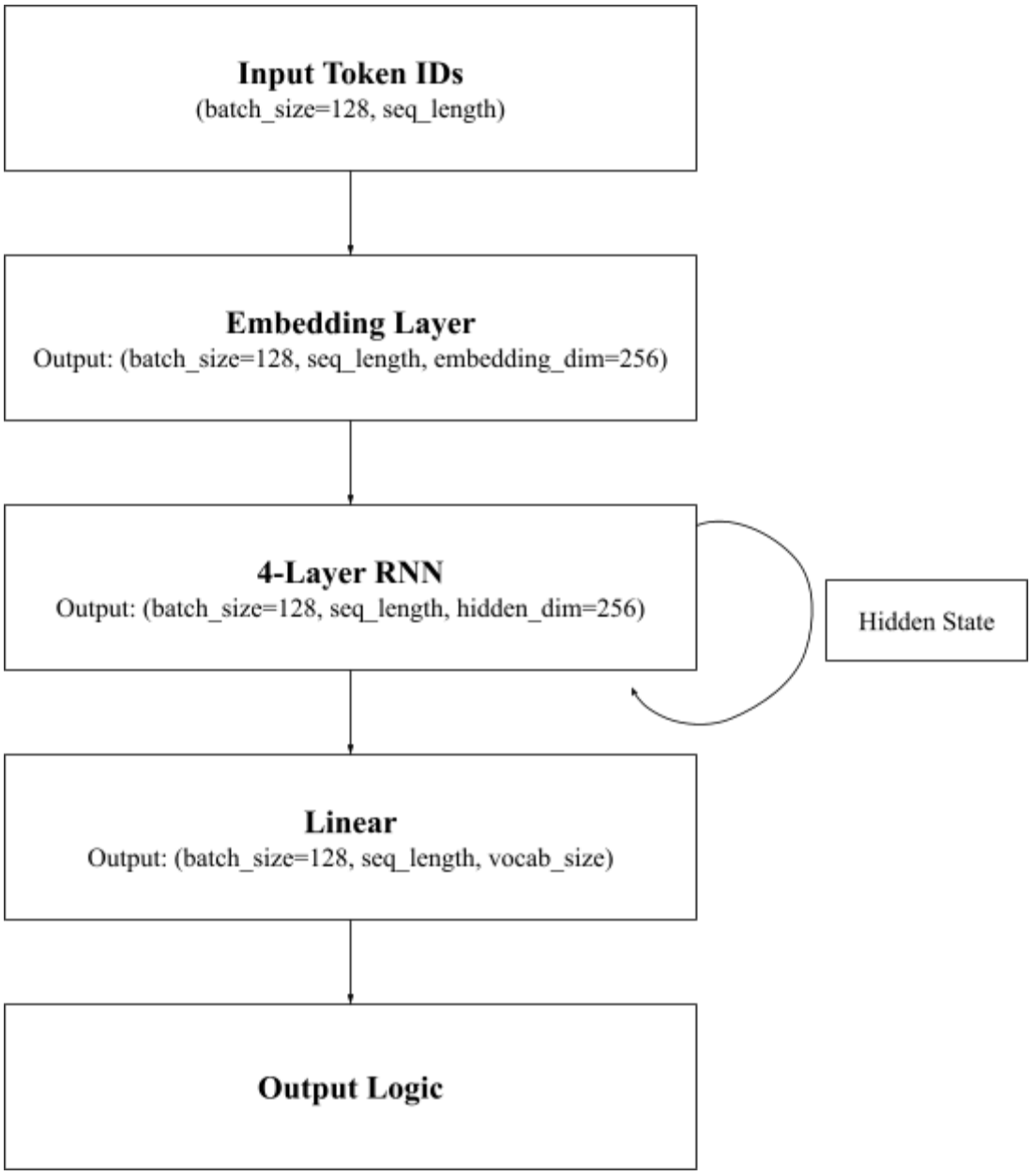
calculates performance metrics using Perplexity (PPL) and BLEU score to evaluate the quality of the generated text. The generate_text.py script accepts a trained model and a user-provided prompt, generating text using either greedy decoding or probabilistic sampling based on the specified temperature. A temperature value of zero triggers greedy decoding, while higher values introduce randomness to the output. Instructions for using these scripts are documented in the project's README file.

The final structure of the project emphasizes modularity and reuse. The root directory contains the core scripts for training, evaluation, and generation, along with a README for guidance. The base_model module serves as the foundation for all model classes and handles shared functionality such as token processing, checkpointing, and output generation. Each model module only needs to define its own __init__ and forward methods, since the rest of the behavior is inherited. The tokenizer script, located in the tokenizer folder, prepares the raw text data, trains the tokenizer, and saves the necessary mappings. The language_dataset module wraps the processed data into a PyTorch-compatible dataset class for efficient loading during training.

Recurrent Neural Network (RNN) Architecture:

I used 4 layers in the final implementation of my RNN architecture. I initially started with 6 layers, but I was curious to see if reducing the depth would lead to better results. After lowering it to 4 layers, I saw improvements in both perplexity and BLEU score, so I kept that configuration for the final model

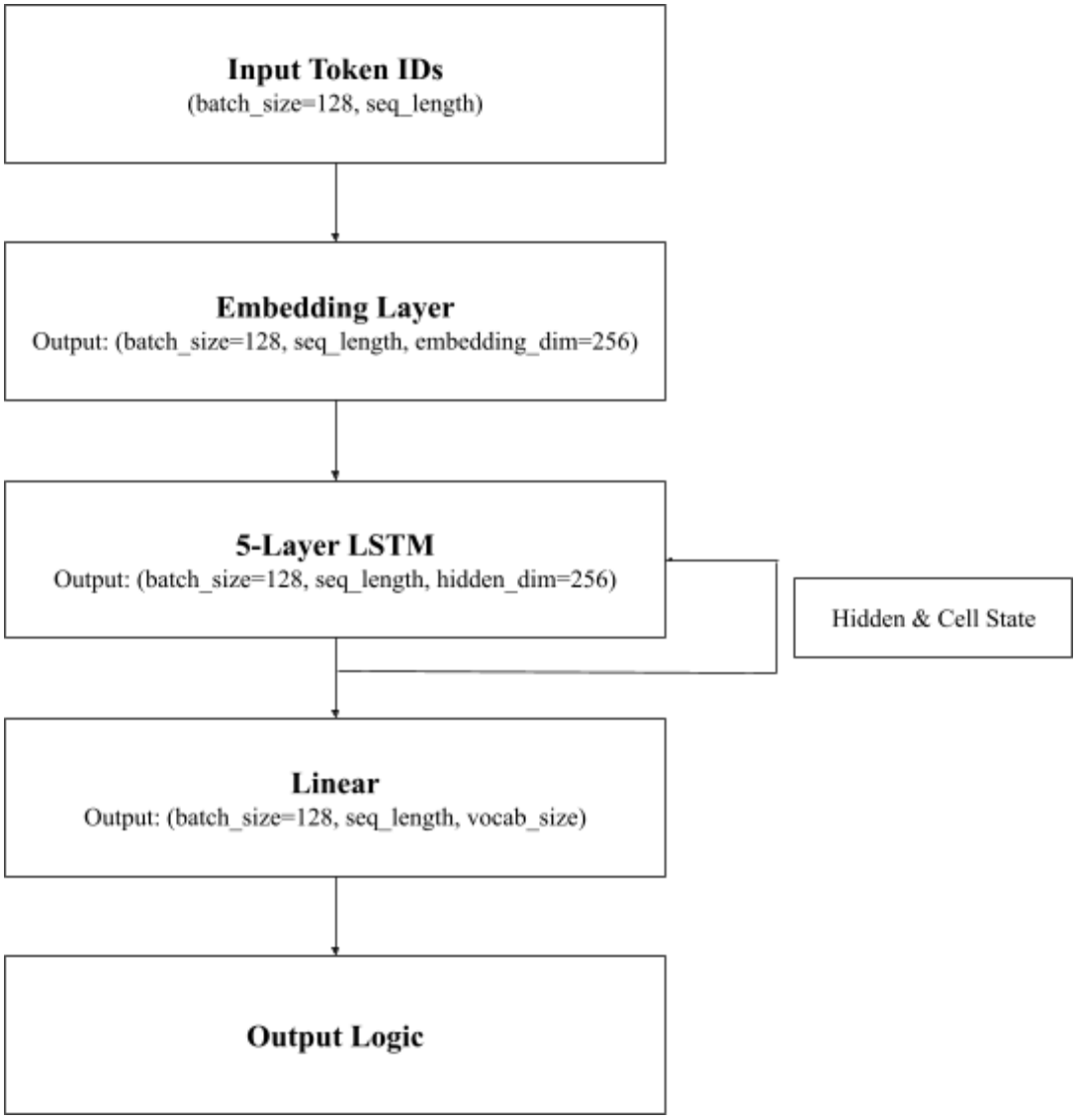
Here is a diagram of my Recurrent Neural Network Architecture:



Long Short-Term Memory (LSTM) Architecture:

I ended up using 5 layers in the final version of my LSTM model. I initially started with 6 layers, thinking it would be a solid baseline for training. But surprisingly, the LSTM actually performed worse than the RNN. After that, I adjusted to 5 layers, which seemed to show slight improvement in overall performance.

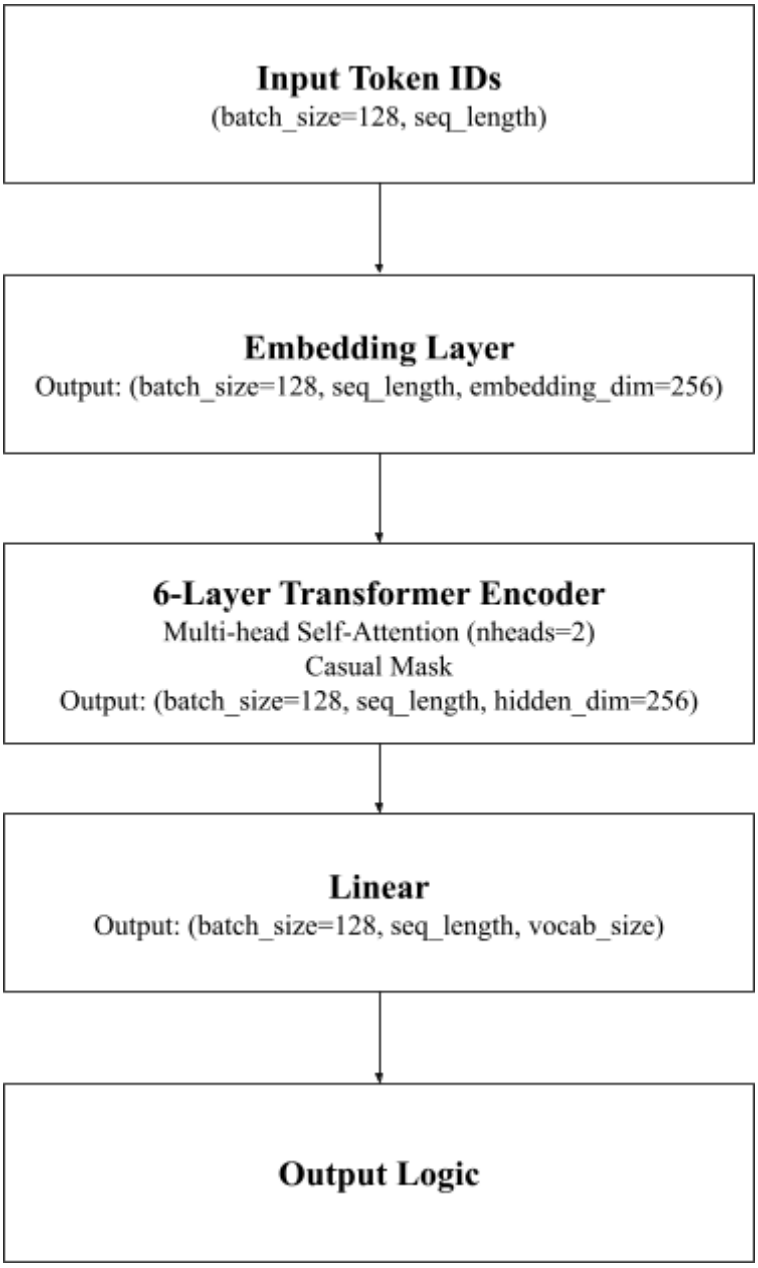
Here is a diagram of my Long Short-Term Memory Architecture:



Transformer Architecture:

I used 6 layers and 2 attention heads during my final implementation of my Transformer architecture. Due to time constraints, I did not train with any other number of layers to compare performance.

Here is a diagram of my Transformer Architecture:



Results

The table below presents the best Perplexity and BLEU score results for each model. As expected, the Transformer model performed the best overall, outperforming both the RNN and LSTM architectures. It achieved the lowest perplexity, indicating stronger confidence in next-token prediction, and the highest BLEU score, showing more fluent and accurate text generation. Surprisingly, the LSTM model performed the worst. Given more time, I would have explored additional ways to improve its performance. However, I did experiment with reducing the number of layers from 6 to 5, which led to some improvement, as explained in more detail below.

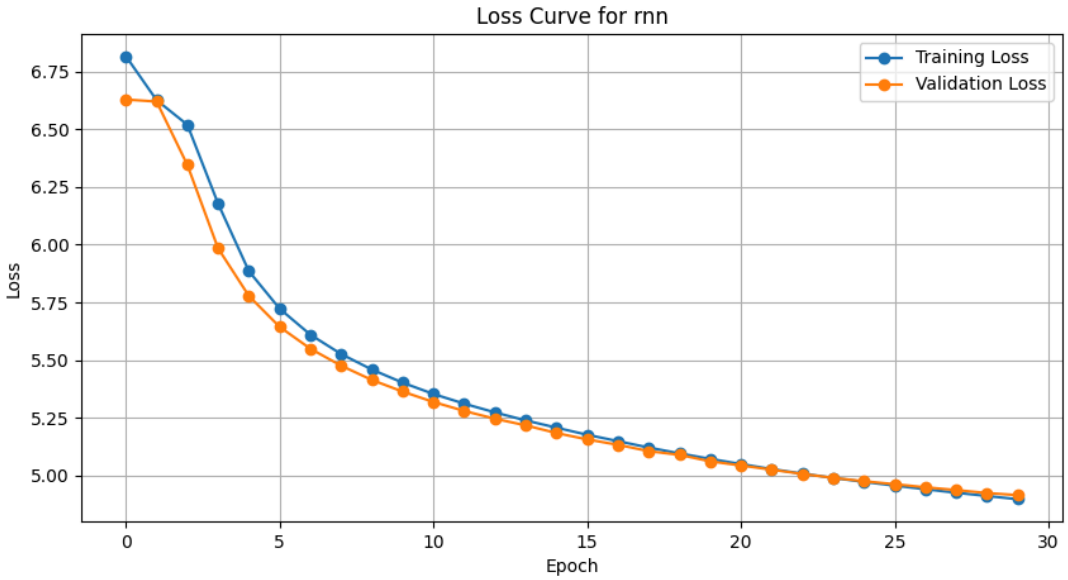
Model	Perplexity (PPL)	BLEU Score
RNN	123.7343	0.0102
LSTM	146.9298	0.0090
Transformer	101.2431	0.0119

For the LSTM model, I was hoping to get better results than the RNN, so I started by using a 6-layer setup like I mentioned earlier. However, it ended up performing worse, with a perplexity of 174.4135 and a BLEU score of just 0.0086. Since that clearly wasn't ideal, I decided to lower the number of layers to 5. That change led to a moderate improvement, and the updated results shown in the table above reflect this 4-layer version of the LSTM.

For the RNN model, I initially used 6 layers and got a perplexity of 136.3725 and a BLEU score of 0.0095. I was curious to see if using fewer layers would make a difference, so I reduced it to 4. Surprisingly, this led to better results, which are reflected in the table above.

***Results of Recurrent Neural Network (RNN)**:*

Here is the loss and validation curve generated while I trained the RNN Model.



For the prompt *"Which do you prefer? Dogs or cats?"*, each model was used to generate a response maxed at 100 tokens, with a sampling temperature set to 0.8.

Here is the terminal command to run this example:

```
python3 generate_text.py --model rnn "Which do you prefer? Dogs or cats?" --max_len 100 --temp=0.8
```

Generated Text:

```
And from some of the only a powerfulmachines so a flight or death, sir. Then I was the letter to the year to keep, I said he gave me. How other, that the most severe debt; whether my father, a swans business then the three minutes that the same, as a single occasion, to possess pass, she did not account of her business at last, when she knew Im ⁇ ntre aristocracy. They lived by the gowly by the
```

For the prompt *"What is a firefighter?"*, each model was used to generate a response capped at 100 tokens, with a sampling temperature set to 1.0.

Here is the terminal command to run this example:

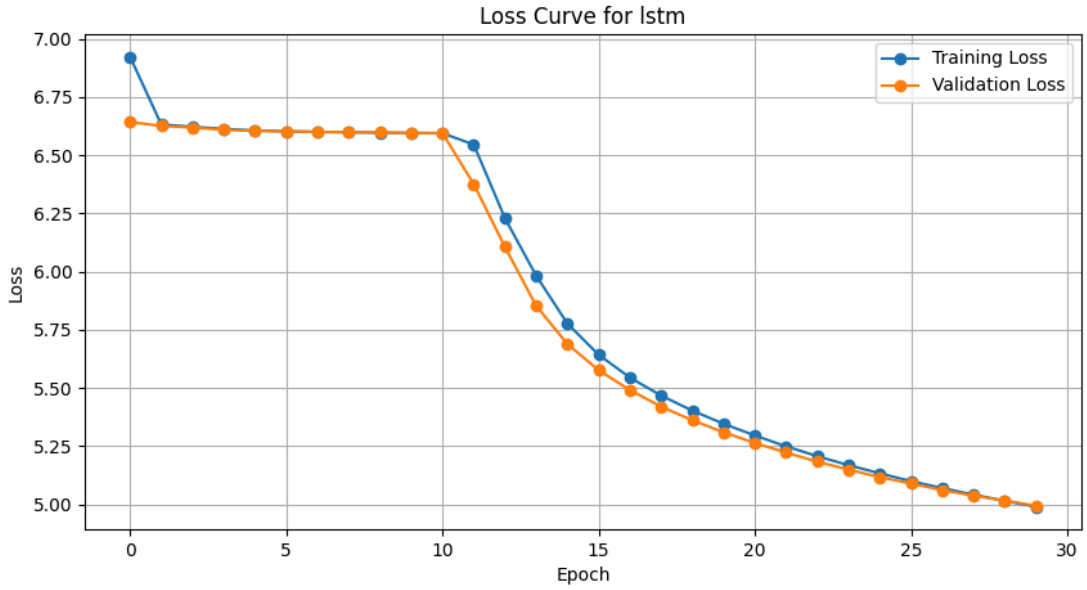
```
python3 generate_text.py --model rnn "What is a firefighter" --max_len 100 --temp=1.0
```

Generated Text:

night about of other way of the other of very secret contrivance within particular for. 27ping; you are changedchaise, Rooted brave dreams, and acrosscast ourselves to the army fixed ups Swthe dog, call you are not at one to the other at the matter only of wine lies, remarked our philosophy; Art and drive at the music curled is yet seem now, he hoped to a man; and called writing, and elevated to permit the greatest contempt and that I forgot. We shall

Results of Long Short-Term Memory (LSTM):

Here is the loss and validation curve generated while I trained the LSTM Model.



For the prompt *"Which do you prefer? Dogs or cats?"*, each model was used to generate a response maxed at 100 tokens, with a sampling temperature set to 0.8.

Here is the terminal command to run this example:

python3 generate_text.py --model lstm "Which do you prefer? Dogs or cats?" --max_len 100 --temp=0.8

Generated Text:

I have the course to sining. recall as he is to you? know so buu Montague can say!... on the book with a elderly, they were sitting in the middle of the man.load to give witness in all present. moreover, a man, garden, and we long in friend in conversation and of the result. muscular, half an most plan. Alas. shining of a good. Five thousand saving without the air and

I saw that be a journeyting, now I well as

For the prompt *"What is a firefighter?"*, each model was used to generate a response capped at 100 tokens, with a sampling temperature set to 1.0.

Here is the terminal command to run this example:

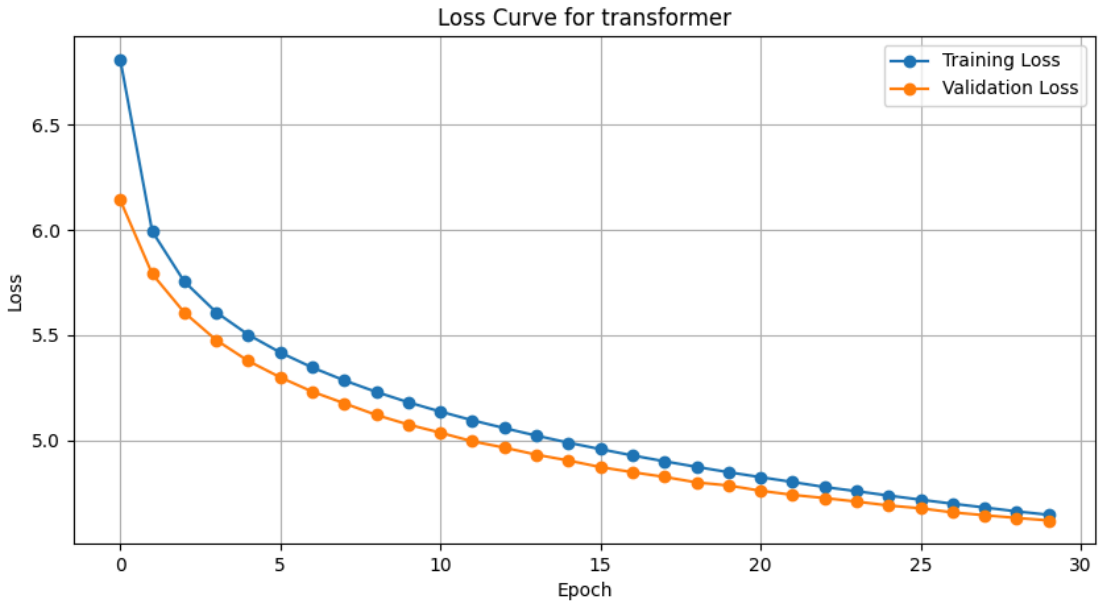
```
python3 generate_text.py --model lstm "What is a firefighter" --max_len 100 --temp=1.0
```

Generated Text:

confidence and mice of equal petticoatfell, to, of spoken to the it do I can be always taken round Brevet, retires With lampwwple. Yes, practise belowsers. press of high a frock that was to affect which indeed she had dine of Americanly are thrown on guardel who are sure and the flowerring with earth own ringsnerable by ranks below, he received the judges you wish, peeres? plump invited the spots, goodness of consideration;

Results of Transformer:

Here is the loss and validation curve generated while I trained the Transformer Model.



For the prompt *"Which do you prefer? Dogs or cats?"*, each model was used to generate a response maxed at 100 tokens, with a sampling temperature set to 0.8.

Here is the terminal command to run this example:

```
python3 generate_text.py --model transformer "Which do you prefer? Dogs or cats?"
--max_len 100 --temp=0.8
```

Generated Text:

To the cabs, are a valiant body had been not have been seen that that the Arch spread on our look attren of the following on the names of the suitorsDe, that her way of whom the counts were not an in the effect onward, and he did not, at the inner and I myself to the illlis of this in the Empire; though he was found a fresh and the ⁇ s. I know who are of the opening! the side, in

For the prompt *"What is a firefighter?"*, each model was used to generate a response capped at 100 tokens, with a sampling temperature set to 1.0.

Here is the terminal command to run this example:

```
python3 generate_text.py --model transformer "What is a firefighter" --max_len 100
--temp=1.0
```

Generated Text:

ies in order would, " ⁇ bo...at lady of them, Rapclet, but he said the ladders on the same company, whether he would not have always because I have discovered by the fifty years ago the purposes, fathers on the Frenchman whether he thought it all was to feed liberty, apparent thoughts is true side the true life of thinking or whom I kept unable to fleeming the rascal but what is established at the household if all direction this the morning. There was

Code Repository Link

https://www.github.com/CatherineRodriquez04/csc4700-project2

Conclusion

Overall, this project helped me build a stronger understanding of older language models that serve as the foundation for the generative models we use today. While these models don't perform at the same level as modern ones, I really enjoyed experimenting with them and seeing the outputs they produced in response to my prompts. It was interesting to observe how each model handled the same input differently. As I mentioned in the Results section, the Transformer

model performed the best, which was expected. The LSTM model had the weakest performance, which could be due to the way I implemented architecture. Still, comparing the models and seeing the differences in their behavior gave me a better appreciation for how these systems work.

This project also gave me the chance to practice writing object-oriented code, which is something I've been wanting to improve, especially as I prepare to begin my PhD in the fall. It also gave me more hands-on experience with PyTorch, which I know will be incredibly useful for future research and projects. Both this project and Project 1 made me feel more confident in my ability to write code that is not only functional but also well-structured and maintainable. Writing the report itself was also a helpful part of the process. It made me slow down and really think through what I had done from start to finish. I sometimes have trouble putting things into words, so this gave me a chance to reflect and express my process more clearly. I genuinely enjoyed this project, not just because of the fun and creative responses from the models, but also because it helped me grow both technically and personally.