




A Deep Learning Sitcom Script Generator
Grace Lee, Jian Cong Loh, Riya Dulepet



INTRODUCTION



The end of a hit sitcom series like *Friends* is invariably followed by calls from fans for a reboot of the show. In our project, we attempt to create a deep learning model to generate new lines for a particular sitcom based on its existing script. Our model is based on OpenAI's Generative Pre-Training Transformer (GPT), a powerful language model which uses a decoder-only transformer architecture and involves pre-training the model on unlabelled text before fine-tuning it for specific downstream tasks.

DATA

Our model is trained on a *Friends* script dataset which contains the script for all episodes of the sitcom. To pretrain the model, we used the WikiText-2 language modelling dataset which contains Good and Featured articles on Wikipedia.

We used HuggingFace's Tokenizer API to build a byte-pair encoding (BPE) tokenizer with a vocabulary size of 10,000. BPE is a subword tokenization technique that allows us to represent rare words while keeping the vocabulary size small by breaking complex words into subtokens.

We used the trained tokenizer to encode both datasets and transform them into appropriate inputs for our model by adding special tokens.



Dataset	Original line	Encoded line
Friends	Rachel: Oh, I'm sorry, did my back hurt your knife?	'[BOS]', 'Rachel', '[DELIM]', 'Oh', ';;', 'I', '""', 'm', 'sorry', ';;', 'did', 'my', 'back', 'hurt', 'your', 'K', 'ni', 'fe', '?', '[EOS]', '[PAD]', ..., '[PAD]'
WikiTex t-2	Rachel Karen Green is a fictional character , one of the six main characters who appear in the American sitcom Friends .	'[BOS]', 'Rachel', 'K', 'aren', 'Green', 'is', 'a', 'fictional', 'character', ';;', 'one', 'of', 'the', 'six', 'main', 'characters', 'who', 'appear', 'in', 'the', 'American', 'sit', 'com', 'Friends', ';;', '[EOS]', '[PAD]', ..., '[PAD]'

Figure 1. Example output of preprocessing. There are ~61k lines for the *Friends* dataset and ~98k lines for the *WikiText-2* dataset after filtering for lines within the window size of 64 tokens.

ARCHITECTURE

We largely replicated GPT's decoder-only transformer architecture, but scaled down the number of parameters.

The input passes through embedding and positional encoding layers of dimension 256, followed by 4 decoder blocks. Each block does masked self-attention with 256 dimensional states and 4 attention heads and has a feed-forward network with 256 dimensional states and an ReLU activation function. We use an Adam optimizer with a learning rate of 0.01 and pretrained and finetuned our model for 1 epoch each with batches of size 64.

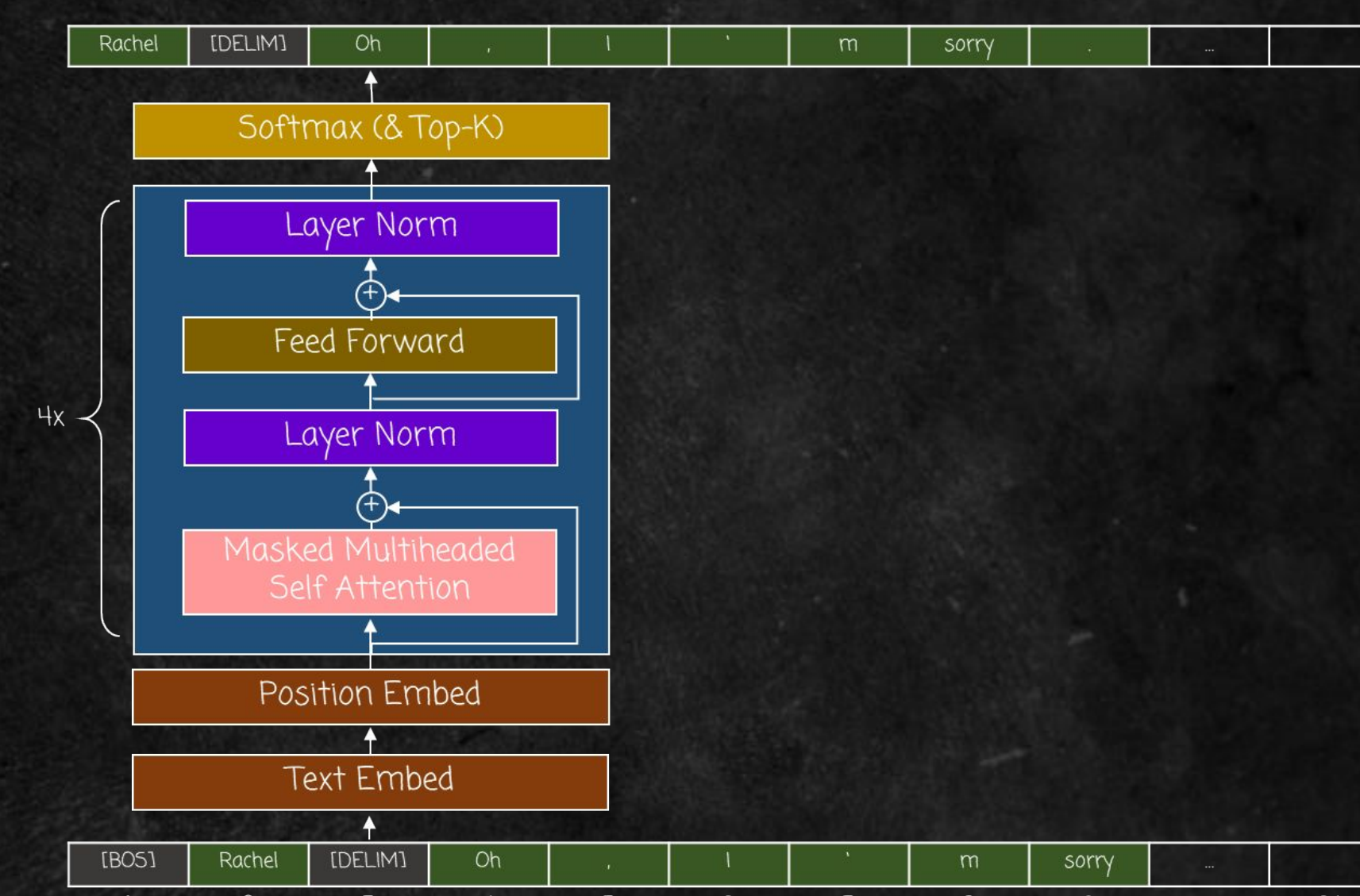
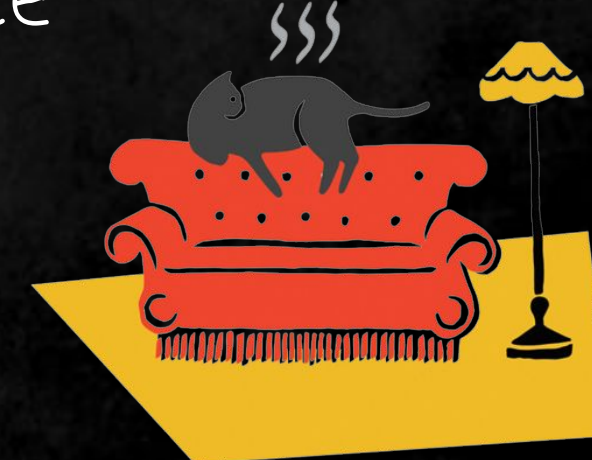


Figure 2. Summary of model architecture. The model has a total of 6.1M parameters

To generate new sentences, we supply the model with an input sequence consisting of a beginning-of-sequence token, a character name, and a delimiter token and start our prediction from the 3rd position. The model outputs a probability distribution over the vocabulary and we randomly select one of the top-k tokens with the highest probabilities, where k is set to 10. This token is added to the input sequence and the process is repeated until we encounter an end-of-sequence token or reach the window size.



RESULTS

Our best model as described in the "Architecture" section achieved a perplexity of 37.4 on the test set for finetuning.



Character	Generated Line
Monica	What ?! I ' m just saying you ' re going ! And the matter , it was a nice guy . It just , he - she was just , you know how I could you say it ? It ' d like you . You ' ll get a little bit more than the other time , you you
Ross	Oh my God . I ' ve got it !

Figure 3. Sample lines generated. Our model is able to generate short sequences that are somewhat coherent, but fail with longer sequences. The generated lines also reflect some common topics and expressions from the show.

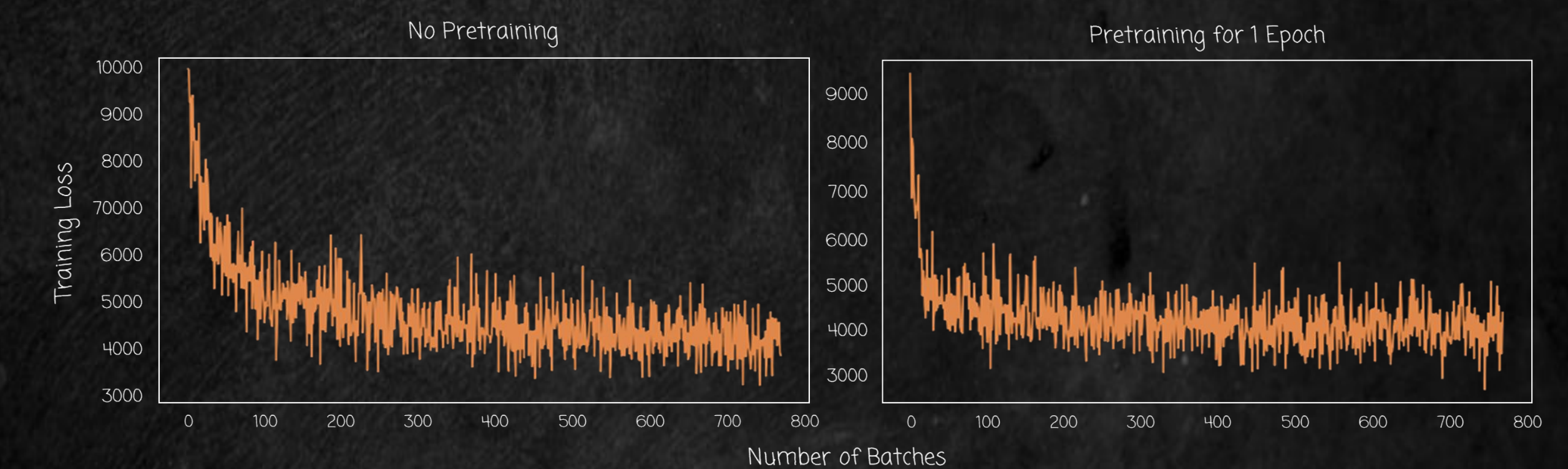


Figure 4. Training loss per batch. Pretraining for 1 epoch gave a lower perplexity compared to no pretraining (37.4 vs 44.5) and led to faster learning.

Total parameters	No pretraining	Pretraining for 1 epoch	Pretraining for 10 epochs
6.1M	44.45	37.38	44.50
17M		328.15	327.55

Figure 5. Perplexity scores for some experiments. Pretraining for more epochs did not lead to better performance, while a larger model performed significantly worse,

Changes to other hyperparameters (learning rate, batch size) and the addition of dropout layers did not yield significant improvements.

DISCUSSION

Generating sitcom lines that are both coherent and semantically resemble the original script is a challenging task that is generative and open-ended. In the original GPT paper, the pretrained model is finetuned for discriminative tasks, which are much more manageable.

The performance of our model was mainly limited by the lack of good data and sufficient computational resources. Given more time and computational resources, we would be able to pretrain the model on a larger and more diverse or relevant dataset. We would also be able to run more experiments with small tweaks to individual model parameters or hyperparameters so that we can Better isolate their effects on the model.

