The Reboot:
A Deep Learning Sitcom Script Generator

Grace Lee (glee73), Jian Cong Loh (jloh4), Riya Dulepet (rdulepet)

1. Introduction

The end of a hit sitcom series like Friends and The Office is invariably followed by calls from fans for a reboot of the show. Therefore, as fans of the sitcom genre, our team decided to create a deep learning model to generate new lines for a particular sitcom based on its existing scripts.

Our project involves language modeling, which is a self-supervised machine learning task. Using deep learning to generate sitcom lines is a complex and interesting problem as we are not only interested in generating natural language, but would like to produce text that semantically resembles lines from a particular sitcom.

Our model is based on OpenAI's Generative Pre-trained Transformer (GPT), a powerful language model which uses a decoder-only transformer architecture and involves pre-training the model on unlabelled text before finetuning it for specific downstream tasks.

2. Methodology

2.1 Data

Our main data comes from the Friends TV Show Script dataset on Kaggle, which contains the script for all episodes of the *Friends* sitcom. In preprocessing, we filtered out lines that contain episode information and stage directions to leave only lines with character dialogues. We also parsed out the character who spoke the line as a separate list.

To pretrain our model, we used the WikiText-2 language modelling dataset which contains Good and Featured articles on Wikipedia. In preprocessing, we removed lines that correspond to headers to leave only the article text and separated each line, which contains a single paragraph in the article, into individual sentences.

To tokenize and prepare our data for the model, we used HuggingFace's Tokenizer API to build and train a byte-pair encoding tokenizer (BPE) with a vocabulary size of 10,000. BPE is a subword tokenization algorithm that starts with all the characters in the corpus as tokens and repeatedly identifies and merges the most common pair of tokens until the vocabulary has reached the specified size. Used in advanced language models like GPT, BPE allows us to represent rare words while keeping the vocabulary size tractable by breaking complex words down into subtokens.

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

- `[BOS]` - beginning of sequence
- `[EOS]` - end of sequence
- `[DELIM]` - delimiter for finetuning
- `[PAD]` - padding

The following shows an example output of the encoding on the two datasets:

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]'`
WikiText-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]'`

After filtering out lines that exceed our window size of 64, we are left with ~61K lines for the *Friends* dataset and ~98 K lines for WikiText-2.

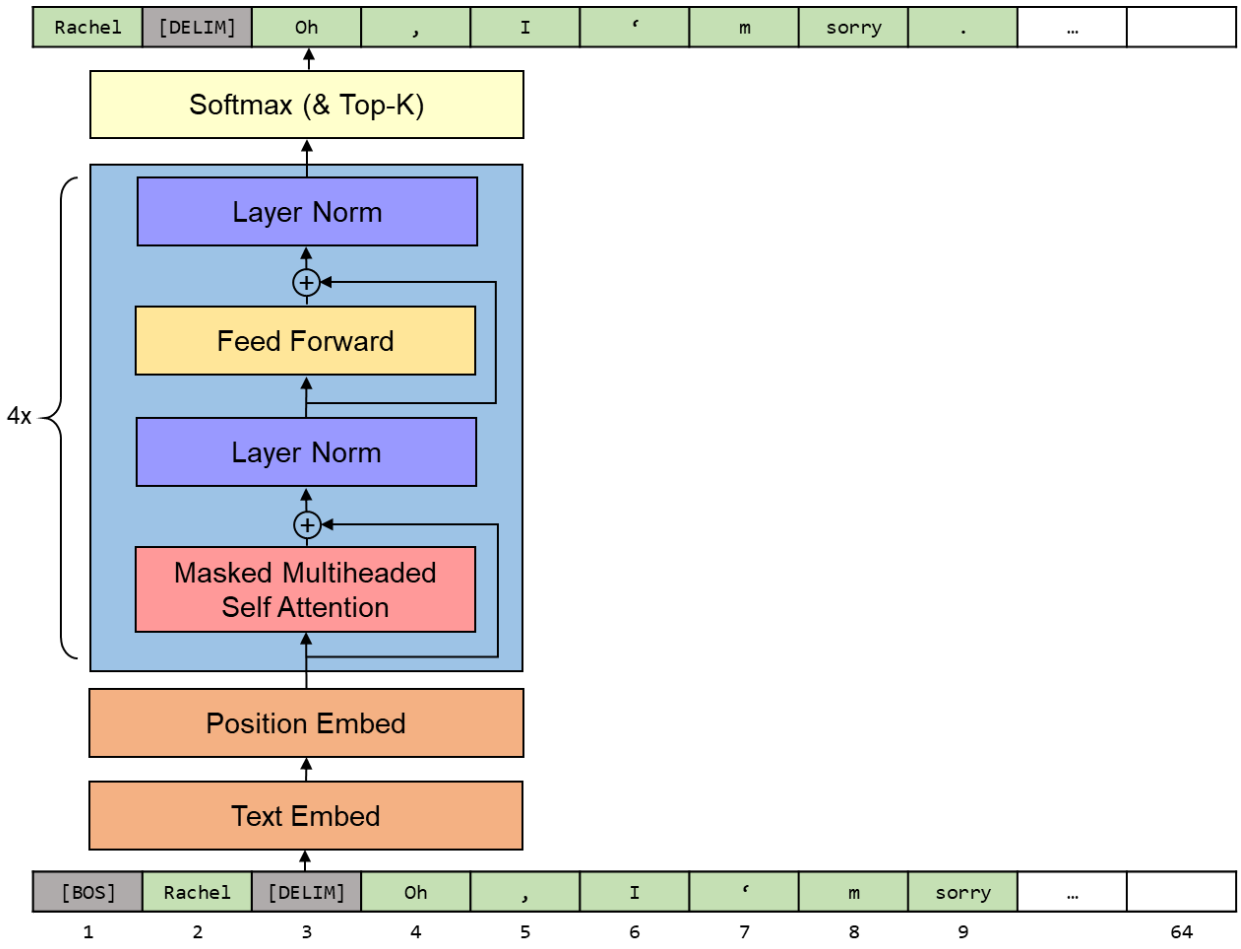
2.2 Model Architecture

We largely replicated the decoder-only transformer architecture of GPT, but scaled down the number of parameters and size of our model. We first passed our inputs through an embedding layer and a learnt positional encoding layer of dimension 256. We then used 4 decoder blocks, each of which does masked self-attention with 256 dimension states and 4 attention heads. For the feed forward network in the decoder block, we used 256 dimensional inner states and an Rectified Linear Unit activation function.

We used the Adam optimizer with a learning rate of 0.01 and pretrained our model for 1 epoch on batches of 64. For finetuning, we trained the model for 1 epoch on batches of 64. Our model has a total of 6.1M parameters.

To generate new sentences once our model is fully trained, we supply the model with an input sequence with a beginning-of-sequence token, a character name, and a delimiter token and start our prediction from the 3rd position (corresponding to the first token after the delimiter). 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.

The figure below summarizes our model architecture:



3. Results

3.1 Results of Best Model

Our best model with the architecture and hyperparameters described above achieved a perplexity of 37.38 on the test set for the finetuning task. The following are some sample lines generated by our trained model:

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
	Oh my god .
Rachel	Well , we just wanted a good idea of my dad , but you know . You ' ll go . And if we ' d have it ' s just , we ' re just gonna go . You don - I just want a good time you , we have any guy . And , I think that .
	I dont have a baby ?
Phoebe	What do not think I ' ve a little bit of my parents , it - it - he just had been in love with a little bit of a good idea of us and you think it was a good time ! I was not the matter ! And he is , that she was a good idea , I
	What are we talking about ? What is that ?
Joey	Hey - hey - hey - look , what did it be a great deal , how did we get it to say you ' d you think I was ? You ' ll get out , and we have to get out , and then , you have any guy , we could just have to go .
	I ' ve to get a little bit . I just want to get a baby !
Chandler	Well I dont think that , we should just go in a while ! Okay ! You are going on the one of a lot . So ? You ' s like you . And I think it ' ll happen !
	Well that is a really great idea .
Ross	I think I have any time ! You ' re gonna go back to my apartment ! I - you ' m sorry you guys are so much , we can have any money ! I mean . I mean it would like , that was like the guy , we were gonna go to me with it with you .
	Oh my God . I ' ve got it !

In general, our model seems to prefer shorter lines and is able to generate short sequences that are somewhat coherent. However, longer lines generated by the model are largelty incoherent and appear to involve stringing short sequences together without taking into account the context of the entire line. This could be a result of our datasets consisting more short sequences such that the model is unable to learn a good representation over a longer sequence of tokens.

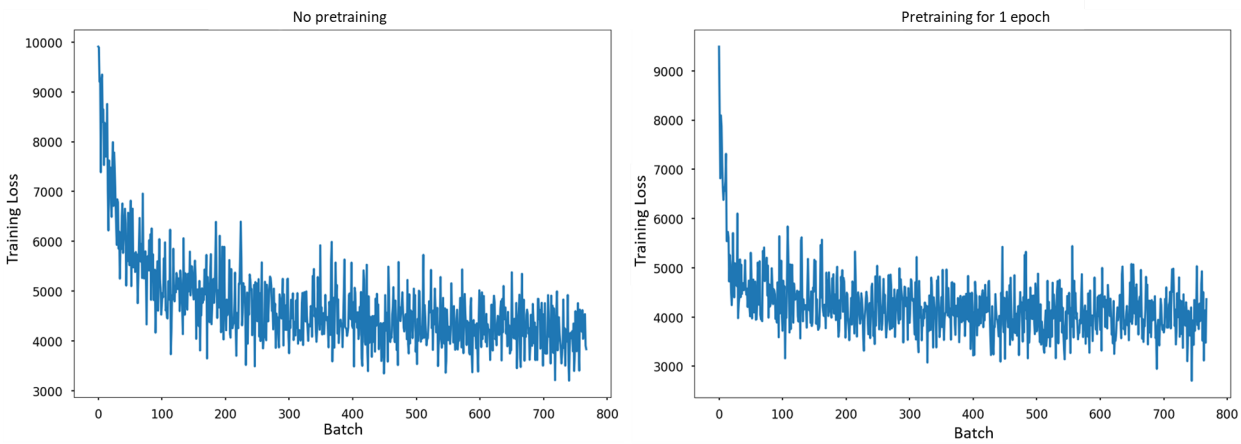
The generated text does seem to somewhat semantically resemble lines spoken by characters in *Friends*. Common topics in the series such as dating and parenthood and frequent expressions of some characters appear occasionally in the generated text.

3.2 Experiments Conducted

We experimented with varying amounts of pretraining and number of parameters, and got the following perplexity scores on our test set:

	No pretraining	**Pretraining for 1 epoch**	**Pretraining for 10 epochs**
Embedding size: 128 No. of decoder blocks: 2 No. of attention heads: 2 **Total parameters: 2.7M**	54.12	43.07	50.90
Embedding size: 256 No. of decoder blocks: 4 No. of attention heads: 4 **Total parameters: 6.1M**	44.45	**37.38**	44.50
Embedding size: 512 No. of decoder blocks: 8 No. of attention heads: 8 **Total parameters: 17M**		328.15	327.55

The core idea of the original GPT paper is that we can improve a model's performance on a specific natural language processing (NLP) task by first doing generative pre-training of a language model on unlabelled text, followed by fine-tuning on the specific task. Indeed, we see that pretraining our model on WikiText-2 allowed us to achieve a better performance on our task. By looking at the training loss per batch, we also see that our model is able to learn faster and converge towards a lower loss more quickly with pretraining:



While increasing the number of epochs of pre-training allows us to stabilize our pretraining loss at a lower value, it does not yield better performance on our specific task. We postulate that this could be a result of the quality of our pretraining dataset, which is not large and diverse enough for our model to learn additional text representation after the first epoch.

Surprisingly, increasing the number of parameters in our model resulted in significantly worse performance on our task and resulted in the model generating a series of punctuations instead of text. Unfortunately, we were unable to determine the exact reasons for this. The losses during both pretraining and finetuning were significantly higher for this model compared to our best model, suggesting that overfitting was unlikely to be the cause. We also suspected that stacking more decoder blocks might have resulted in a vanishing gradient problem, but the same issue persisted when we kept the number of decoder blocks at 4 and only increased the embedding size and number of attention heads. Our best guess of why this might have occurred is that our data is again not rich enough for a more powerful model to learn meaningful text representations.

We also experimented with tweaking other hyperparameters such as the learning rate and batch size and adding dropout layers for regularization. Unfortunately, none of the changes yielded any improvement to our model.

4. Challenges

The main challenges that we faced in our project has to do with the lack of good data and sufficient computational resources.

We had trouble finding a clean and well-formatted sitcom script dataset, and had to do significant preprocessing to clean the dataset that we eventually settled upon as it had inconsistent formatting across different episodes.

The original GPT model was pretrained on a large and diverse corpus of unlabelled text. To improve our pretraining, we tried to train our model on the significantly larger WikiText-103 dataset which is 181MB, compared to WikiText-2 which is 4.3MB. Our program, however, was repeatedly killed during preprocessing, likely due to the process running out of RAM on our machine. The performance of our model was also likely limited by the lack of diversity of our pretraining dataset which contains only Wikipedia articles. Time constraints, however, prevented us from collecting a more diverse or relevant dataset.

We also faced significant challenges in tweaking our model architecture and finetuning our hyperparameters to create a better-performing model. Given the large number of model parameters and hyperparameters and time and resource constraints, it was difficult for us to experiment individually with many small changes to isolate and determine their effects on the model. Our model also suffered from the lack of interpretability. We plotted the batch losses for each run to help us with diagnosing the problems with our model but were unable to gather much from the visualizations.

5. Reflection

As mentioned previously, generating sitcom lines is a two-fold problem: producing natural language and text that semantically resembled script lines. In retrospect, even our base goal of generating a single sitcom line that is both coherent and semantically sound is overly ambitious given the inherent difficulty of language modelling. In GPT, the pretrained model is finetuned for specific discriminative tasks such as classification, answering multiple choice questions, and determining similarity between different sequences. The specific task in our project, however, is still generative in nature and therefore significantly more open-ended and challenging. A more tractable problem that we could have considered, for instance, would be to determine the character given a particular line from the sitcom. Nonetheless, our model did produce interesting results, generating lines that contain coherent short sequences and that to a small extent semantically resembles the lines from *Friends*.

We initially planned to only replicate the GPT architecture without doing pretraining, but realized that the pretraining process was an important contributor to GPT's state-of-the-art performance. Pretraining did indeed allow us to improve the performance of the model on our specific task.

Given more time and computational resources, we would be able to address the challenges discussed in the previous section. Specifically, we would be able to pretrain our model on a larger and more diverse dataset and conduct more experiments with different model parameters and hyperparameters to determine the best model.

Overall, the project enabled us to gain a deep understanding into GPT and the process of pretraining and finetuning, as well as work with new techniques like BPE. The project also enabled us to learn about the process of deep learning research, particularly in designing and conducting experiments.