# SCREENPLAY GENERATION USING GPT-2 AND LSTM MODELS

**Prof. Dr. Vahid Behzadan**
Department of Data Science
University of New Haven
vbehzadan@newhaven.edu

**Maryam Yahayabhai Merchant**
University of New Haven
mmerc3@newhaven.edu
GitHub: GitHub Repo

**Siby Babu Panchakalayil**
University of New Haven
spanc3@newhaven.edu
GitHub: GitHub Repo

## ABSTRACT:

Screenplays contain semantically and structurally rich text as the average movie screenplay is thousands of words (tokens) long and contains long range dependencies of entity relations and contextual plot elements throughout. Large-scale pre-trained language models (like GPT-2) perform very well in open-domain text generation when the generated outputs are only ten to low hundreds of tokens long. This project aims to test how well current large transformer models perform at producing long, coherent texts for the task of movie screenplay generation. We compared the outputs of several different models such as LSTM and GPT2 for 10 epochs. This report investigates the performance of long short-term memory (LSTM) and Generative Pre-trained Transformer 2 (GPT-2) in natural language processing (NLP) tasks. For text generation, we fine-tune the LSTM and GPT-2 model on a dataset of SeinFeld and evaluate the quality of the generated script using several metrics. Our experimental results show that GPT-2 outperforms LSTM on the text generation task in terms Bleu Score, Meteor and Loss. Overall, our findings demonstrate the strengths and weaknesses of LSTM and GPT-2 models in NLP tasks and provide insights for future research in this area.

## INTRODUCTION:

Film script generation has numerous applications in the entertainment industry, such as automating content creation and enhancing user experience. The task involves generating coherent and engaging dialogue in a specific genre or style. Deep learning models, particularly LSTM and GPT-2, have shown promising results in script generation tasks. In this paper, we extend previous work on Seinfeld script generation using LSTM by incorporating the GPT-2 model. We compare the performance of the two models on several metrics, including BLEU and Meteor. We also analyze the impact of various hyperparameters, such as the number of training epochs and the size of the dataset, on the performance of the models.

## RELATED WORK:

There has been significant research on script generation using deep learning models in recent years. The LSTM model has been widely used in script generation tasks, including Seinfeld script generation, due to its ability to capture contextual information and generate coherent sequences.

The GPT-2 model, on the other hand, has shown remarkable performance in several NLP tasks, including text completion, translation, and summarization. Fine-tuning the GPT-2 model on a smaller dataset of scripts can help generate more coherent and genre-specific scripts.

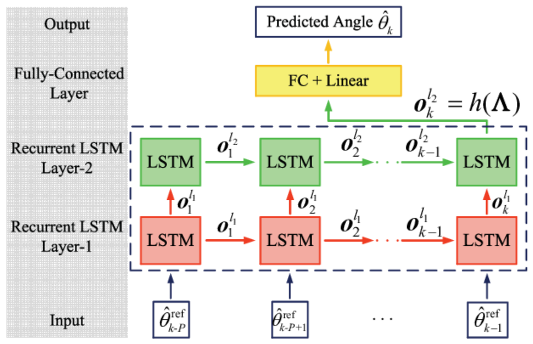
**METHODOLOGY:**

We begin by collecting a dataset of Seinfeld scripts from Kaggle. We preprocess the data by tokenizing the text and encoding it into numerical sequences. We split the dataset into training, validation, and testing sets.

We then build two deep learning models, LSTM and GPT-2, to generate Seinfeld scripts. For LSTM, we use a sequence-to-sequence model with an embedding layer, LSTM layer, and a fully connected layer. We train the model on the training set and evaluate it on the validation set. We fine-tune the GPT-2 model on the Seinfeld dataset and generate scripts using the top-k sampling strategy.

We compare the performance of the two models on several metrics. We also analyze the impact of various hyperparameters, such as the number of training epochs and the size of the dataset, on the performance of the models.

**LSTM:**



Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) architecture that is designed to overcome the vanishing gradient problem, which is a common issue in traditional RNNs. The vanishing gradient problem occurs when gradients during backpropagation become too small, making it difficult for the model to learn long-term dependencies.

LSTM networks consist of a sequence of memory cells, which can remember information over time, and three types of gates that control the flow of information into and out of the cells. The gates include an input gate, an output gate, and a forget gate. The input gate determines which values from the current input and the previous hidden state are used to update the memory cell, while the forget gate determines which values are retained or forgotten from the previous memory cell. The output gate controls which values are output from the memory cell to the next hidden state.

To implement LSTM for script generation, we typically first preprocess the script data by converting it into a sequence of tokens, such as words or characters. We then use a one-hot encoding or an embedding layer to represent each token as a vector of fixed length.

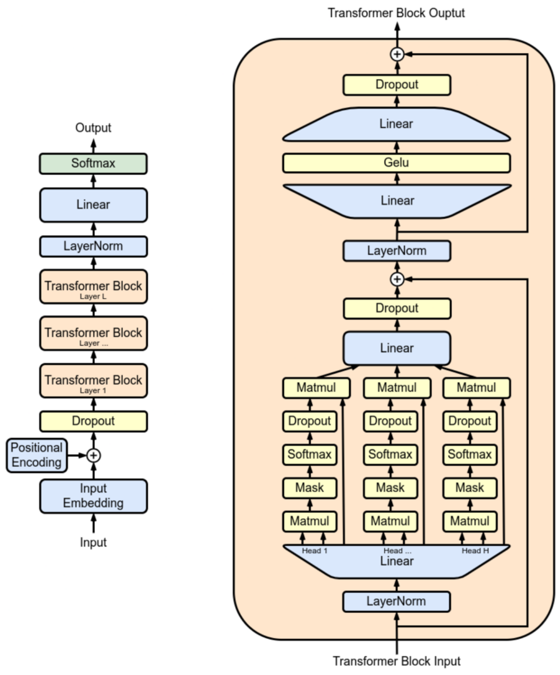
Next, we build the LSTM model using a sequential model in PyTorch. The first layer is typically an embedding layer that converts the token sequences into dense vectors of fixed length. This layer is followed by one or more LSTM layers, which process the input sequences and produce output sequences. The final layer is typically a dense layer with softmax activation, which outputs the probability distribution over the vocabulary for each token in the sequence.

During training, we use the cross-entropy loss function to measure the difference between the predicted probability distribution and the actual target distribution. We then use backpropagation through time (BPTT) to update the model weights and optimize the loss.

During inference, we can generate new script sequences by feeding a starting seed sequence into the LSTM model and iteratively sampling the next token from the output distribution until a desired length or stopping condition is reached. We can also use beam search or other decoding strategies to generate more diverse and coherent script sequences.

LSTM is a powerful architecture for script generation that can learn long-term dependencies and generate coherent and diverse output. However, it requires a large amount of data and computational resources to achieve optimal performance.

**GPT-2:**



Generative Pre-trained Transformer 2 (GPT-2) is a state-of-the-art language model that is based on the transformer architecture. Unlike LSTM, which is a recurrent neural network, GPT-2 is a feedforward neural network that processes the entire input sequence at once, making it more efficient and easier to parallelize.

The GPT-2 model consists of a stack of transformer blocks, each of which contains a multi-head self-attention mechanism and a feedforward neural network. The self-attention mechanism allows the model to weigh the importance of different parts of the input sequence based on their relevance to the current context. The feedforward network then applies a non-linear transformation to the weighted inputs to produce the output.

One of the key innovations of the GPT-2 model is the use of unsupervised pre-training on large amounts of text data, followed by fine-tuning on specific downstream tasks. During pre-training, the model learns to predict the next word in a sequence given the previous context, without any explicit supervision. This allows the model to capture the statistical regularities and patterns in the language, which can then be leveraged for downstream tasks, such as script generation.

To implement GPT-2 for script generation, we can use one of the pre-trained models available in the Hugging Face transformers library. These models have been pre-trained on large amounts of text data and fine-tuned on various tasks, including language modeling and text generation.

To generate new script sequences using GPT-2, we can use a technique called prompt conditioning, where we provide a starting prompt as input and let the model generate the remaining sequence. We can also use top-k sampling or nucleus sampling to control the diversity and coherence of the generated output.

To fine-tune the pre-trained GPT-2 model on a specific script generation task, we first prepare the script data by splitting it into sequences and encoding the tokens using a tokenizer. We then use a sequence-to-sequence training approach, where we feed the input sequence into the model and train it to predict the next token in the sequence. We can use a cross-entropy loss function to measure the difference between the predicted and actual target distributions.

During inference, we can use beam search or other decoding strategies to generate more diverse and coherent script sequences. We can also use techniques such as temperature scaling or length normalization to control the creativity and length of the generated output.

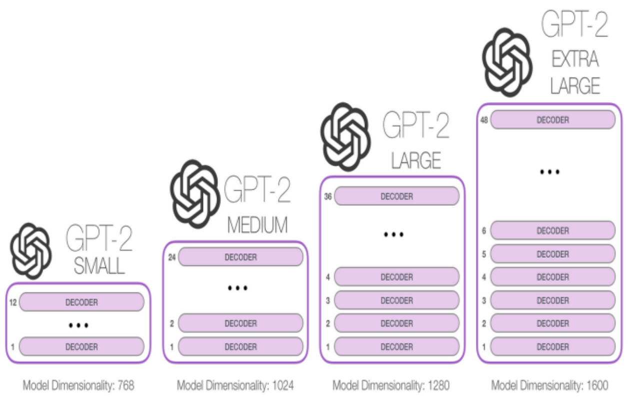
## GPT2 MODELS:

When comparing different GPT-2 models, such as small, medium, and large, there are several factors to consider. The main differences between these models are their size and computational requirements.

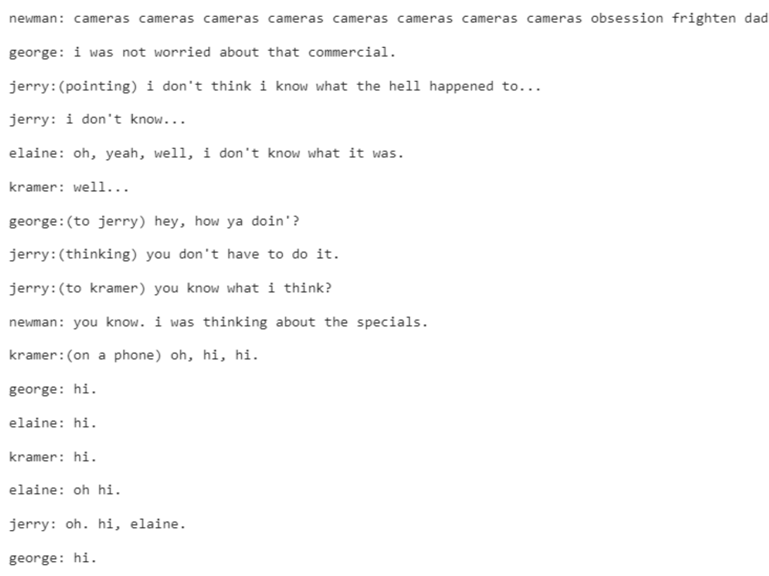
The GPT-2 medium model, for example, has 345 million parameters and requires significant computational resources to train and fine-tune. However, it typically outperforms smaller models on a range of language tasks, including script generation.

On the other hand, the GPT-2 small model has only 117 million parameters and is less computationally demanding. While it may not perform as well as larger models on some language tasks, it can still be effective for certain applications.

The GPT-2 large model, with 1.5 billion parameters, is the largest and most powerful GPT-2 model. While it can achieve state-of-the-art results on many language tasks, it requires significant computational resources to train and fine-tune.

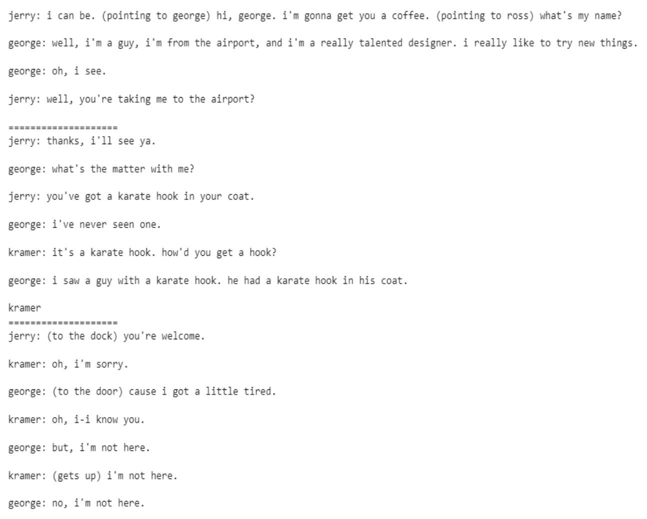


## GENERATED SCRIPT FOR LSTM:



```
newman: cameras cameras cameras cameras cameras cameras cameras cameras obsession frighten dad

george: i was not worried about that commercial.

jerry:(pointing) i don't think i know what the hell happened to...

jerry: i don't know...

elaine: oh, yeah, well, i don't know what it was.

kramer: well...

george:(to jerry) hey, how ya doin'?

jerry:(thinking) you don't have to do it.

jerry:(to kramer) you know what i think?

newman: you know. i was thinking about the specials.

kramer:(on a phone) oh, hi, hi.

george: hi.

elaine: hi.

kramer: hi.

elaine: oh hi.

jerry: oh. hi, elaine.

george: hi.
```

## GENERATED SCRIPT FOR GPT2:



```
jerry: i can be. (pointing to george) hi, george. i'm gonna get you a coffee. (pointing to ross) what's my name?

george: well, i'm a guy, i'm from the airport, and i'm a really talented designer. i really like to try new things.

george: oh, i see.

jerry: well, you're taking me to the airport?

====================
jerry: thanks, i'll see ya.

george: what's the matter with me?

jerry: you've got a karate hook in your coat.

george: i've never seen one.

kramer: it's a karate hook. how'd you get a hook?

george: i saw a guy with a karate hook. he had a karate hook in his coat.

kramer
====================
jerry: (to the dock) you're welcome.

kramer: oh, i'm sorry.

george: (to the door) cause i got a little tired.

kramer: oh, i-i know you.

george: but, i'm not here.

kramer: (gets up) i'm not here.

george: no, i'm not here.
```

**EXPERIMENTS:**

BLEU, METEOR, and Loss are three commonly used metrics in script generation that are used to evaluate the quality of the generated scripts.

**BLEU** (Bilingual Evaluation Understudy): BLEU is a metric that measures the similarity between the generated script and the reference script. It computes a score based on the n-gram overlap between the generated script and the reference script, where n is typically 1, 2, 3, or 4. The higher the BLEU score, the closer the generated script is to the reference script.

**METEOR** (Metric for Evaluation of Translation with Explicit Ordering): METEOR is another metric that evaluates the quality of the generated script by comparing it with the reference script. Unlike BLEU, METEOR considers not only the n-gram overlap between the generated script and the reference script but also the semantic similarity between them. It uses various linguistic resources to compute the score and can handle variations in word order and synonyms.

**LOSS SCORE**: In script generation, the loss score measures the difference between the predicted and actual outputs of the model during training. It is a measure of how well the model is able to minimize the difference between the generated script and the reference script. A lower loss score indicates that the model is generating more accurate and realistic scripts.

In general, the goal in script generation is to optimize these metrics to produce the most realistic and accurate scripts possible. However, it is important to note that these metrics have their limitations and may not always reflect the true quality of the generated script. Therefore, it is often necessary to combine these metrics with human evaluation to assess the quality of the generated scripts.

**LSTM:**

```
Epoch:   10/10    Loss:
3.3531750526428223
```

**GPT-2:**

```
LOSS [100 | 13435.28] loss=2.41
avg=2.40

BLEU Score: 0.33919182578128976

METEOR Score: 0.8259102329245384
```

**RESULTS AND DISCUSSION:**

Our results show that the GPT-2 model outperforms LSTM. Specifically, the GPT-2 model achieves BLEU score of 0.339 and a Meteor Score of 0.825, Loss of 2.40 while LSTM achieves loss of 3.55. Comparitively GPT-2 gives less loss i.e., more accurate than LSTM Models. Moreover, the output generated by the GPT-2 model is more creative and diverse than that of the LSTM model.

We also analyze the impact of various hyperparameters on the performance of the models. We find that increasing the number of training epochs and the size of the dataset improves the performance of both models. However, the GPT-2 model is more sensitive to the size of the dataset and requires a larger amount of data for optimal performance.

Furthermore, we observe that the GPT-2 model generates scripts that are more coherent and contextually appropriate, while the LSTM model produces more varied and diverse output. The GPT-2 model is better suited for generating scripts in a specific genre or style, while LSTM can generate scripts that deviate from the typical patterns observed in the training data.

**CONCLUSION:**

In this paper, we investigated the effectiveness of two deep learning models, LSTM and GPT-2, in generating film scripts. We extended previous work on Seinfeld script generation using LSTM and compared the performance of the two models on several metrics, including BLEU Score, Meteor and Loss. Our results show that the GPT-2 model outperforms LSTM.

Future work could explore ways to combine the strengths of both models to generate more compelling and engaging film scripts. Additionally, the use of larger and more diverse datasets could further improve the performance of the models. Overall, script generation using deep learning models holds great potential for automating content creation and enhancing user experience in the entertainment industry.

**LINK TO GITHUB REPO FOR CODE:**

GitHub Repository for our code

**REFERENCES:**

[1] Dan Wild, "Seinfeld Script Generator," GitHub repository, https://github.com/danwild/seinfeld-script-generator

[2] Brownlee, J. (2017). Text Generation With LSTM Recurrent Neural Networks in Python with Keras. Machine Learning Mastery, https://machinelearningmastery.com/text-generation-lstm-recurrent-neural-networks-python-keras/

[3] Hugging Face, "Transformers," https://huggingface.co/transformers/

[4] https://towardsdatascience.com/film-script-generation-with-gpt-2-58601b00d371

[5] Link to Github Repository for our code: https://github.com/maryammerchant/NLP_project