COMP SCI 5563 - Assignment 4 Report
Group № 3

Group members

Jayadithya Nalajala
MS student in Comp-Sci

Mohammadreza Akbari Lor
PhD student in Comp-Sci

Namuun Lkhagvadorj
MS student in Data Science

Krishnasai Bharadwaj Atmakuri
MS student in Data Science

Jeet Das
PhD student in Comp-Sci

Parisha Rathod
MS student in Data Science

The Art of Automated Poetry

Abstract

The purpose of this project is to delve into the capabilities of artificial intelligence in creative writing by utilizing machine learning techniques to generate poetry. By meticulously analyzing diverse datasets comprising works from multiple celebrated poets, our objective is to develop and fine-tune a machine learning model capable of emulating the unique stylistic elements of these poets. This project not only seeks to harness AI's potential in mirroring human-like creativity but also aims to enhance our understanding of the complex dynamics between computational algorithms and artistic expression. Specifically, the project will explore Large Language Models (LLMs) and their efficacy in capturing and reproducing the nuanced linguistic and thematic characteristics inherent in the works of different poets. Through this exploration, the project aspires to contribute significantly to the field of natural language processing, demonstrating how advanced technologies can be leveraged to expand the boundaries of literature and poetry.

1. Introduction

The fusion of artificial intelligence with creative arts has opened new avenues for exploring the capabilities of machine learning in artistic domains. This project specifically investigates the potential of neural networks, particularly Generative Pre-trained Transformer 2 (GPT-2), in the field of poetry generation. By fine-tuning a pretrained GPT-2 model, this research aims to uncover how advanced language models can be adapted to emulate the poetic styles of various authors and generate novel poetic compositions.

Central to our investigation is the use of a pre-trained GPT-2 model, renowned for its efficacy in a wide range of natural language processing tasks. The decision to leverage a pretrained model lies in its extensive training on a diverse corpus, enabling it to grasp complex language patterns and nuances. This project extends the model's capabilities by fine-tuning it on a meticulously curated dataset comprising poems from multiple poets known for their distinctive styles and thematic diversity. Through this approach, we aim to harness the pretrained model's foundational knowledge and adapt it to the specific task of poetry generation.

The objectives of this project are multifaceted: Firstly, to fine-tune the pretrained GPT-2 model on a specialized corpus of poetic texts, thereby aligning it more closely with the stylistic and thematic attributes of the target poets. Secondly, to evaluate the model’s ability to generate poems that not only reflect the learned styles but also resonate with thematic depth and emotional expressiveness characteristic of human-authored poetry.

This initiative also explores broader implications, questioning the extent to which artificial intelligence can contribute to creative processes. It challenges the conventional boundaries of machine capabilities, offering insights into the intersection of technology and human creativity. The project not only tests the technical and creative boundaries of AI in the arts but also contributes to ongoing dialogues about the nature of creativity and the innovative potential of machine learning in artistic expressions.

In summary, this project utilizes a sophisticated approach by fine-tuning a GPT-2 model to explore AI-driven poetry generation. It stands as a testament to the evolving relationship between artificial intelligence and literary creativity, highlighting both the challenges and possibilities inherent in this exciting interdisciplinary field.

2. Data preprocessing:

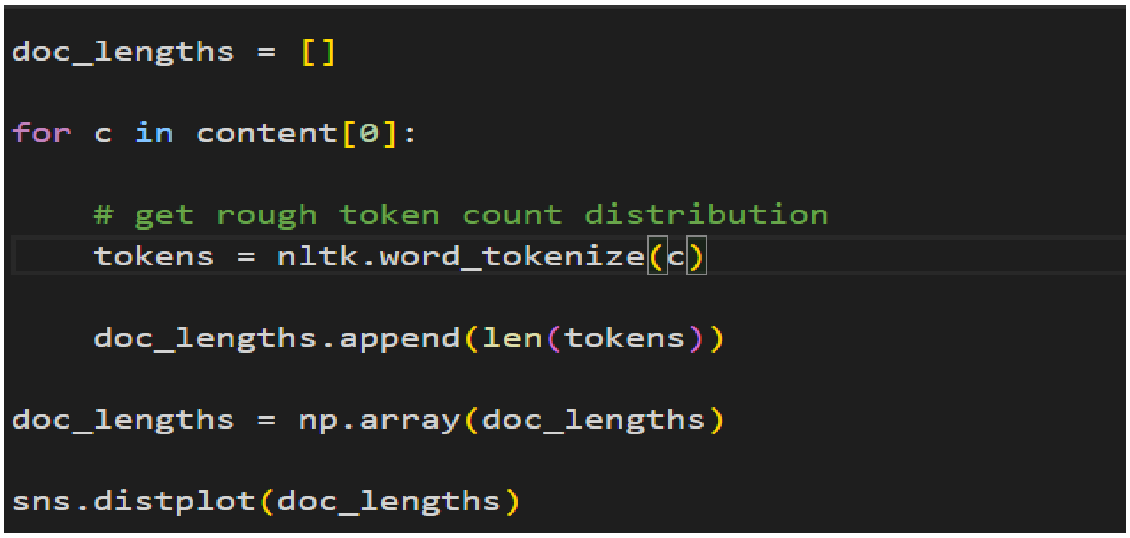
Dataset

We have chosen the following poet datasets for this project.

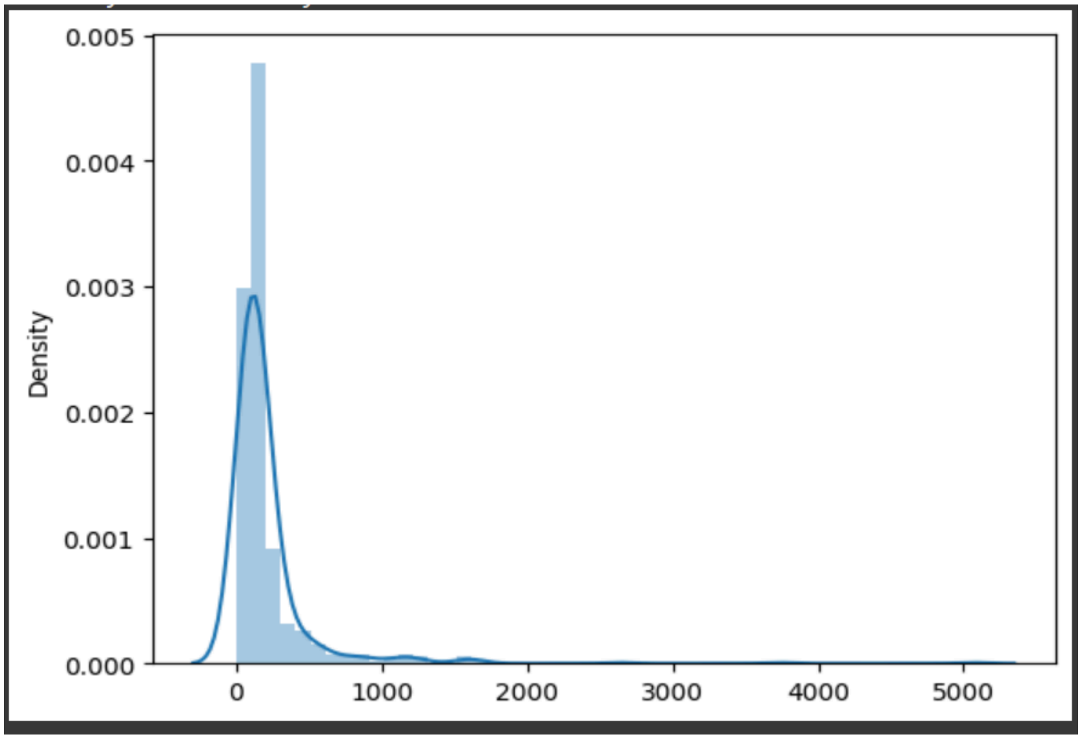
1. https://huggingface.co/datasets/merve/poetry This dataset has 573 unique poems total and are written in english. It contains poems from subjects: Love, Nature and Mythology & Folklore that belong to two periods namely Renaissance and Modern. The dataset has 5 columns such as Content, Author, Poem name, Age, Type.

Document Length Distribution Analysis:

To gain insights into the structure and variability of the poetry dataset, we conducted an analysis of the distribution of document lengths. The following Python code snippet was used to calculate and visualize this distribution:

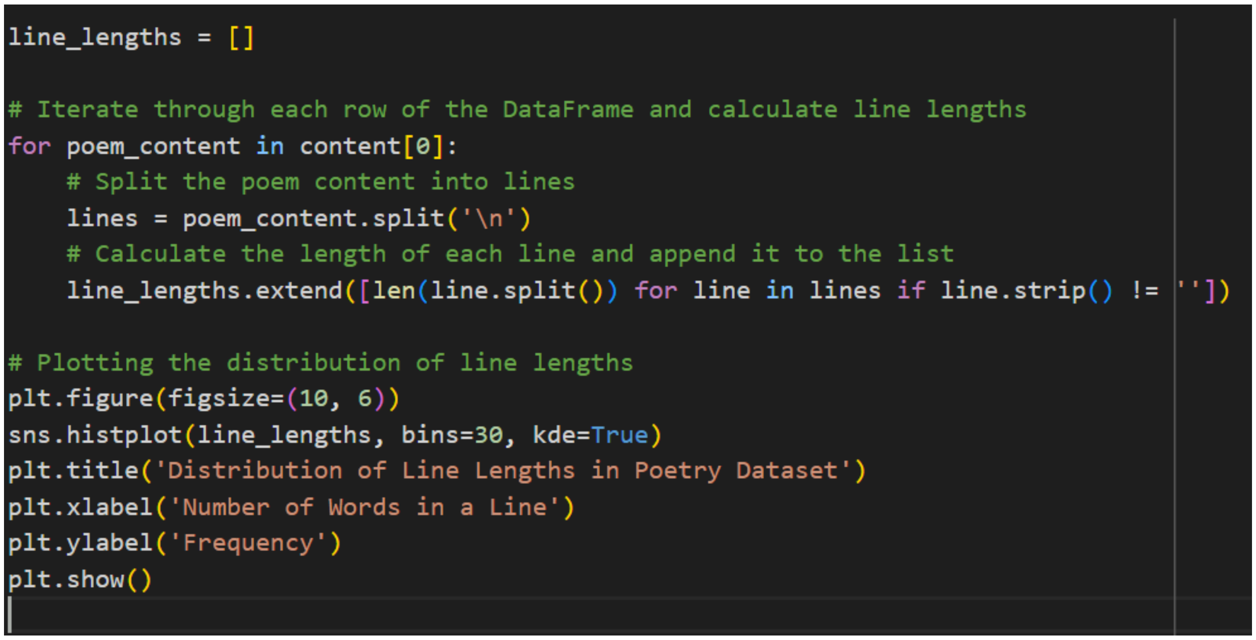


```
doc_lengths = []

for c in content[0]:

    # get rough token count distribution
    tokens = nltk.word_tokenize(c)

    doc_lengths.append(len(tokens))

doc_lengths = np.array(doc_lengths)

sns.distplot(doc_lengths)
```

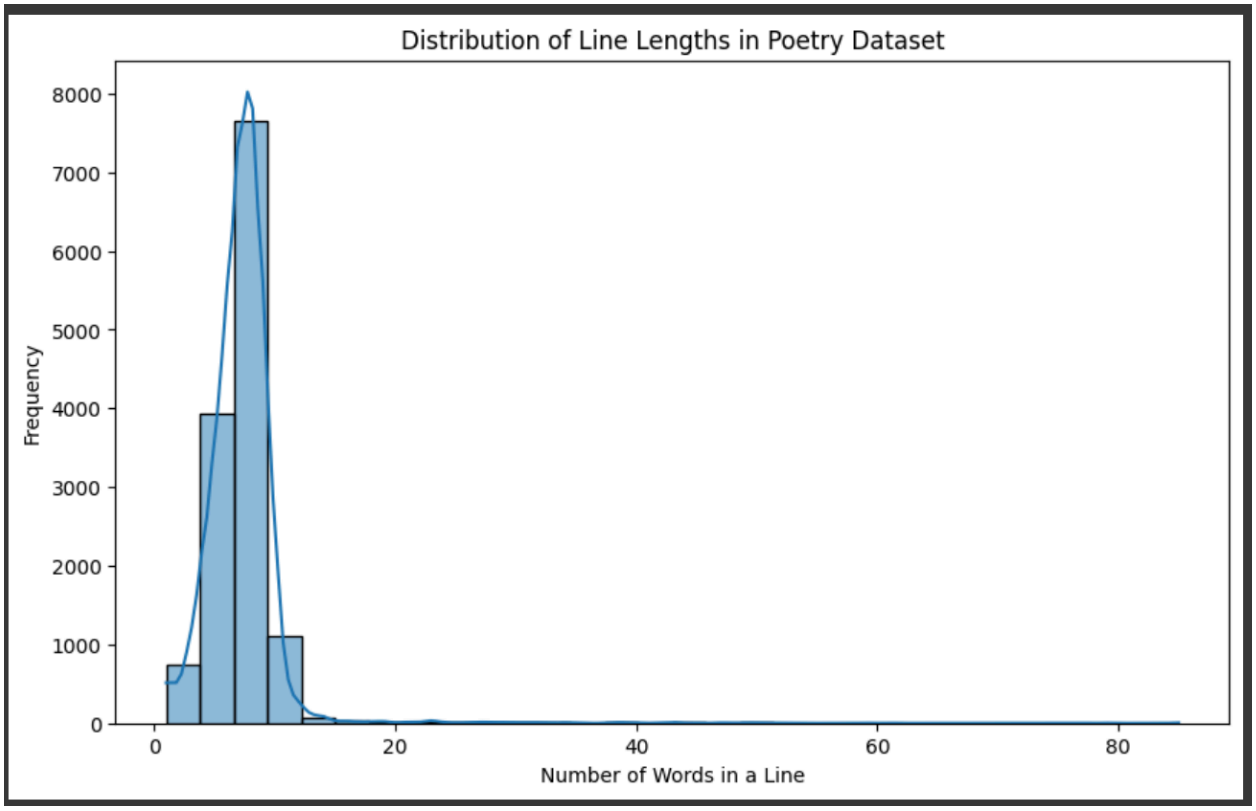


Line Length Distribution Analysis:

In addition to analyzing the distribution of document lengths, we conducted an analysis of the distribution of line lengths within the poetry dataset. The following Python code snippet was utilized to perform this analysis:

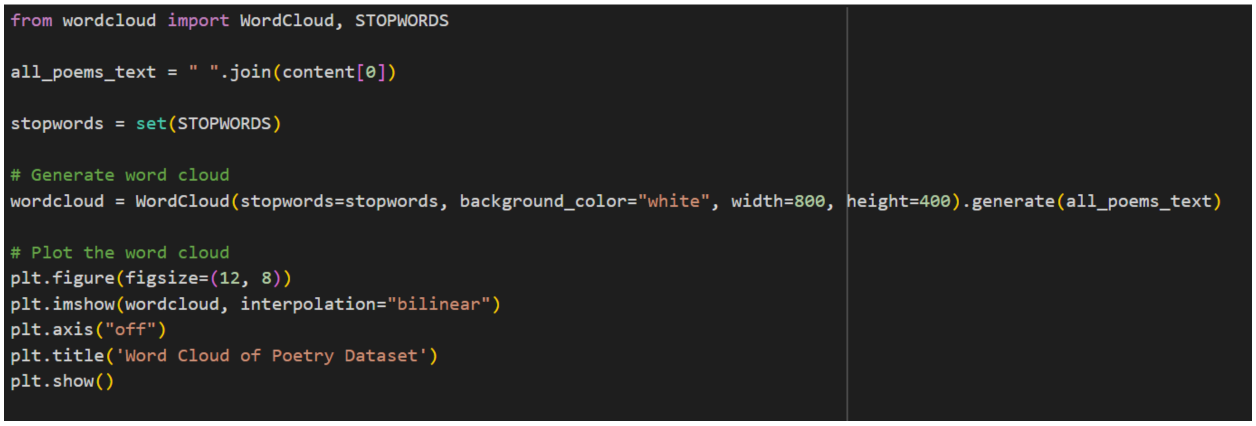


```
line_lengths = []

# Iterate through each row of the DataFrame and calculate line lengths
for poem_content in content[0]:
    # Split the poem content into lines
    lines = poem_content.split('\n')
    # Calculate the length of each line and append it to the list
    line_lengths.extend([len(line.split()) for line in lines if line.strip() != ''])

# Plotting the distribution of line lengths
plt.figure(figsize=(10, 6))
sns.histplot(line_lengths, bins=30, kde=True)
plt.title('Distribution of Line Lengths in Poetry Dataset')
plt.xlabel('Number of Words in a Line')
plt.ylabel('Frequency')
plt.show()
```

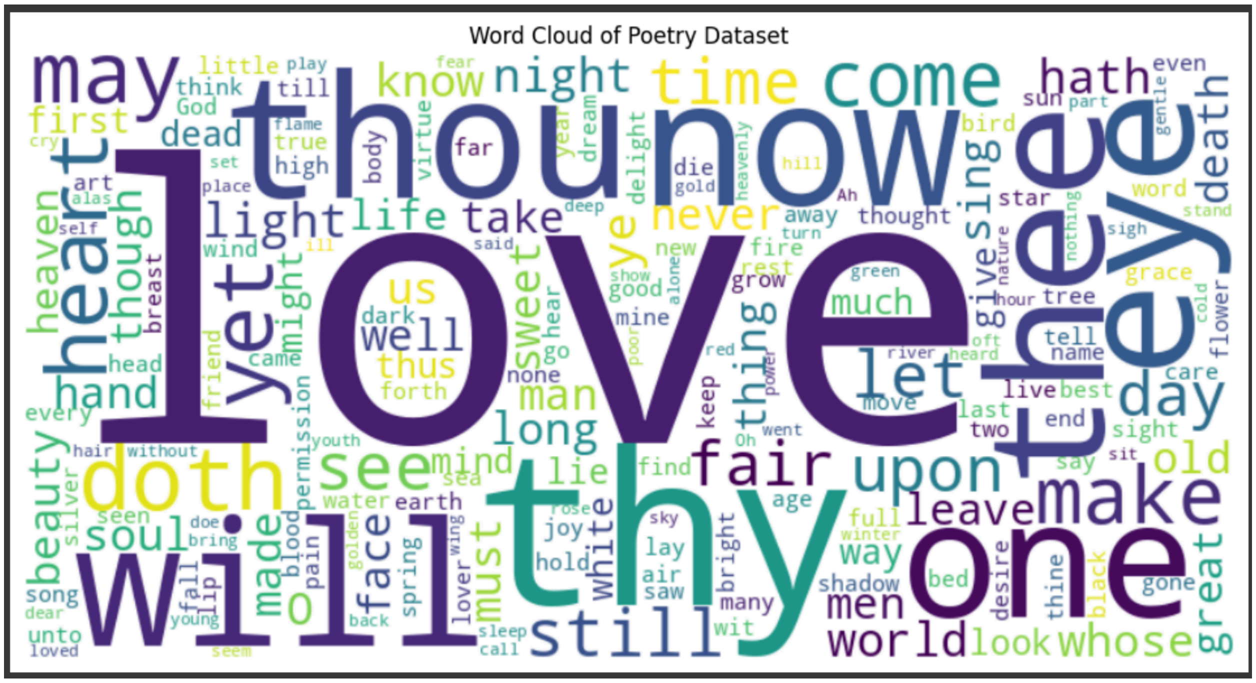


Word Cloud of dataset:

To visually represent the most common words present in the poetry dataset, we generated a word cloud using the WordCloud library in Python. The following code snippet demonstrates how this visualization was created:



```python
from wordcloud import WordCloud, STOPWORDS

all_poems_text = " ".join(content[0])

stopwords = set(STOPWORDS)

# Generate word cloud
wordcloud = WordCloud(stopwords=stopwords, background_color="white", width=800, height=400).generate(all_poems_text)

# Plot the word cloud
plt.figure(figsize=(12, 8))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.title('Word Cloud of Poetry Dataset')
plt.show()
```



The resulting word cloud offers a visual summary of the most frequent words found across the entire poetry dataset.
By examining the word cloud, we can quickly identify prominent themes or topics prevalent in the poetry collection.

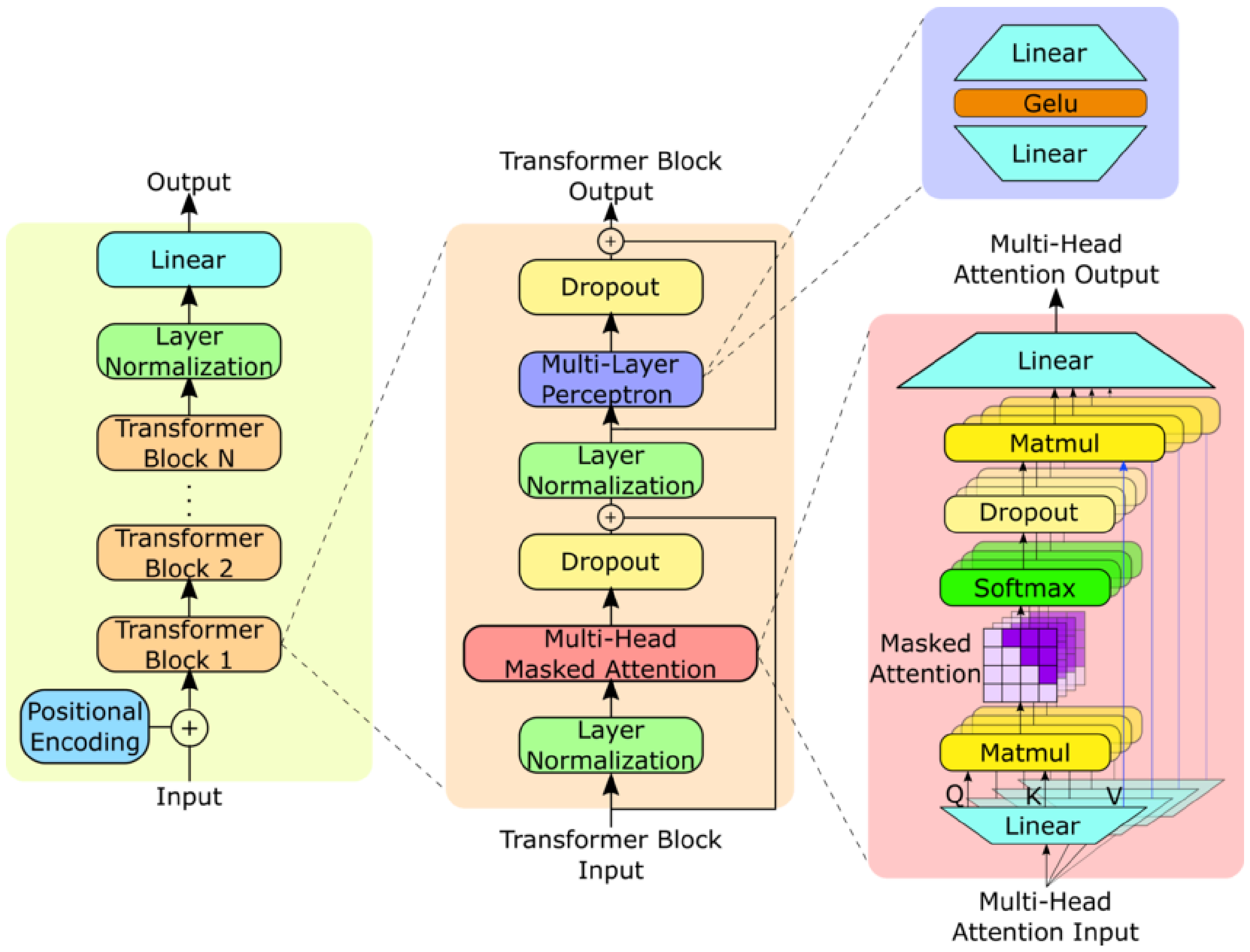
Dataset Preparation:

The dataset used for training was prepared through a custom class *GPT2Dataset*, which takes a list of text and the tokenizer as input. Each text string was preprocessed by adding special tokens (BOS and EOS) and then tokenized. The tokenizer converted texts into sequences of integers and created attention masks to indicate to the model which tokens should be focused on and which are paddings.

This prepared dataset was split into training and validation sets, with 90% of the data used for training and the remaining 10% for validation. This split helps in evaluating the model's performance on unseen data, ensuring that the model generalizes well and isn't just memorizing the training data.

3. Model Description:

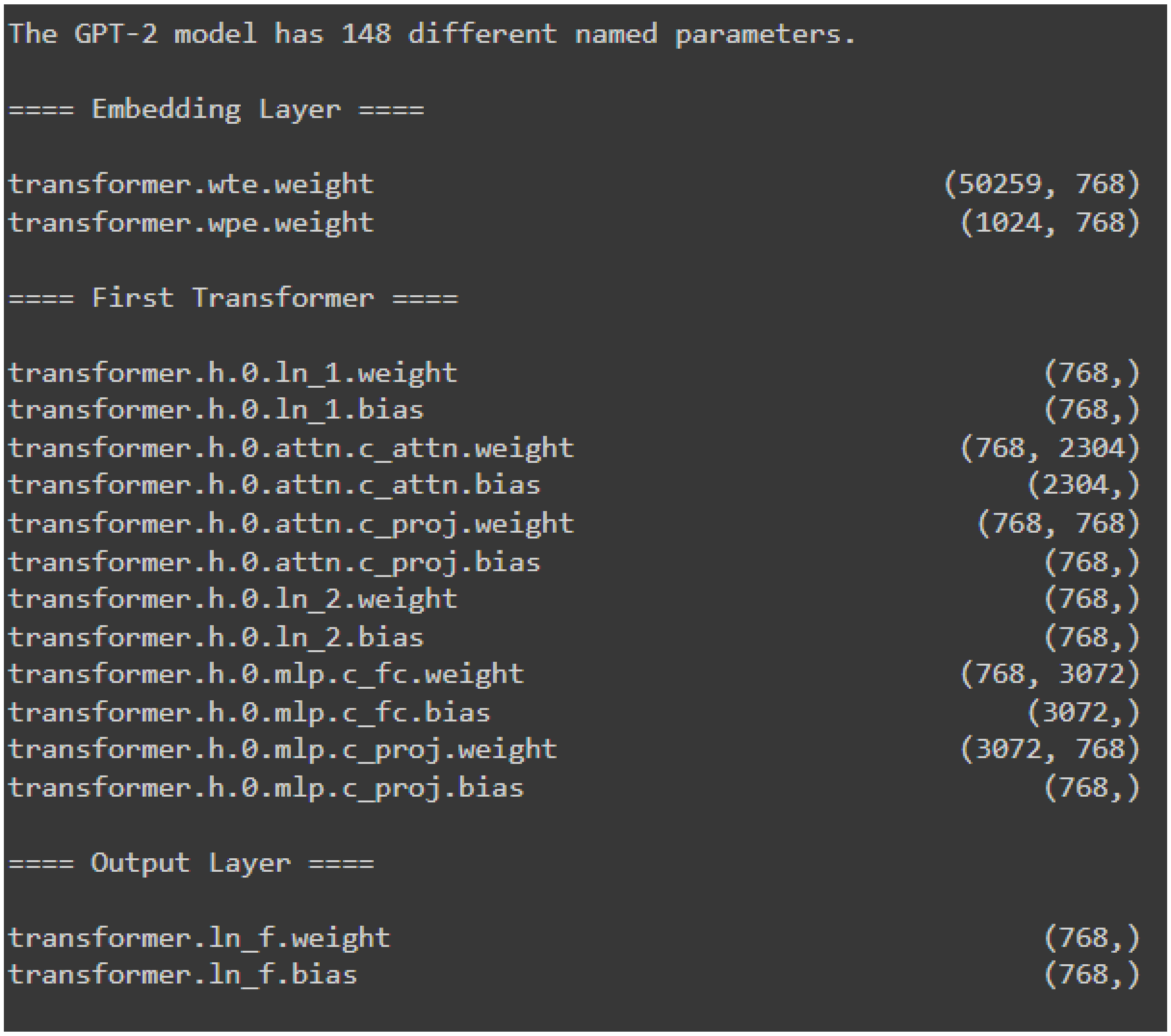
The GPT-2 medium model features a transformer-based architecture with 345 million parameters. This places it between the smaller 124 million and the larger 1.5 billion parameter models in the GPT-2 family, balancing computational efficiency with powerful generative capabilities. Like other GPT-2 variants, the medium model was trained on the "WebText" dataset, which consists of a wide range of internet text. It uses unsupervised learning through language modeling—predicting the next word in a sequence based on the previous words. The GPT-2 medium model exhibits robust performance in generating coherent and contextually relevant text. Its capabilities extend to various natural language processing tasks such as translation, summarization, and question answering, often without the need for task-specific tuning. The GPT-2 medium model is particularly well-suited for applications that require a balance between performance and resource efficiency. It has been used in academic research, content generation, and as a base for further model customization and refinement.



GPT2Tokenizer:

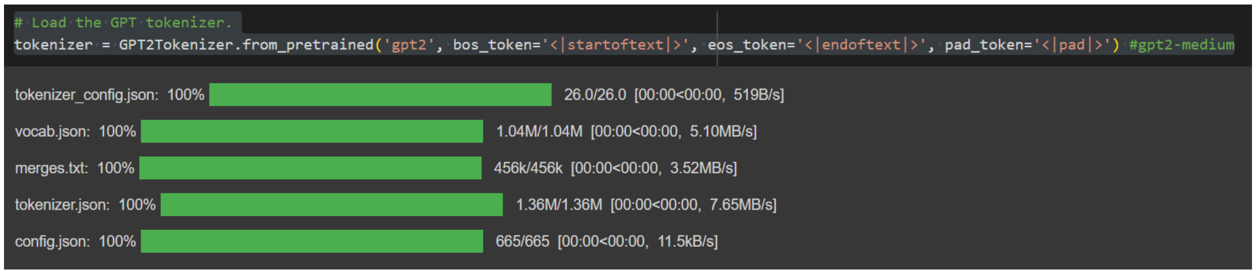
The GPT2Tokenizer is a powerful tool for natural language processing tasks, especially when working with models like GPT-2. It simplifies the process of converting text data into a format suitable for input to the model.

Below is the GPT-2 model summary with the architecture and number of layers along with the number of parameters:



```
The GPT-2 model has 148 different named parameters.

==== Embedding Layer ====

transformer.wte.weight                                    (50259, 768)
transformer.wpe.weight                                     (1024, 768)

==== First Transformer ====

transformer.h.0.ln_1.weight                                     (768,)
transformer.h.0.ln_1.bias                                       (768,)
transformer.h.0.attn.c_attn.weight                         (768, 2304)
transformer.h.0.attn.c_attn.bias                               (2304,)
transformer.h.0.attn.c_proj.weight                          (768, 768)
transformer.h.0.attn.c_proj.bias                                (768,)
transformer.h.0.ln_2.weight                                     (768,)
transformer.h.0.ln_2.bias                                       (768,)
transformer.h.0.mlp.c_fc.weight                            (768, 3072)
transformer.h.0.mlp.c_fc.bias                                  (3072,)
transformer.h.0.mlp.c_proj.weight                          (3072, 768)
transformer.h.0.mlp.c_proj.bias                                 (768,)

==== Output Layer ====

transformer.ln_f.weight                                         (768,)
transformer.ln_f.bias                                           (768,)
```

The model is being loaded in the Colab:



```
# Load the GPT tokenizer.
tokenizer = GPT2Tokenizer.from_pretrained('gpt2', bos_token='<|startoftext|>', eos_token='<|endoftext|>', pad_token='<|pad|>') #gpt2-medium
```

```
tokenizer_config.json: 100%  26.0/26.0 [00:00<00:00, 519B/s]
vocab.json: 100%  1.04M/1.04M [00:00<00:00, 5.10MB/s]
merges.txt: 100%  456k/456k [00:00<00:00, 3.52MB/s]
tokenizer.json: 100%  1.36M/1.36M [00:00<00:00, 7.65MB/s]
config.json: 100%  665/665 [00:00<00:00, 11.5kB/s]
```

GPT2Tokenizer offers various built-in techniques for tokenization, including:

- Word Tokenization: Breaking text into words or subwords.

- Special Tokens Handling: Adding special tokens like <BOS> (beginning of sequence), <EOS> (end of sequence), and <PAD> (padding) tokens.
- Padding: Ensuring all sequences have the same length by padding shorter sequences with special tokens.
- Subword Tokenization: Handling out-of-vocabulary words by breaking them into smaller sub words or characters.
- Vocabulary: Maintaining a vocabulary of tokens, enabling conversion between tokens and numerical indices.
- Tokenization Options: Providing options for different tokenization strategies, such as byte-level encoding, word-level encoding, etc.

Training:

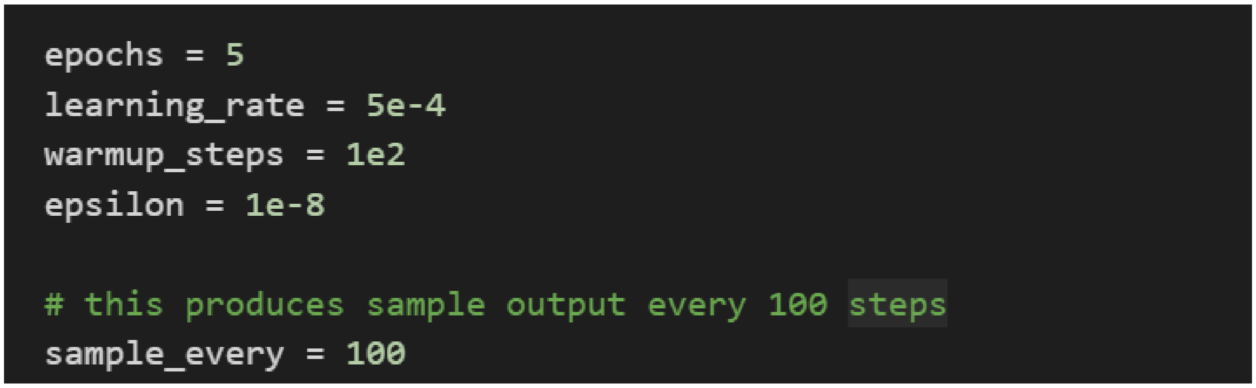
Training a language model like GPT-2 involves configuring several parameters to effectively guide the learning process. These parameters include the number of training epochs, learning rate, and the choice of loss function. The configuration used in this project is designed to optimize the model's ability to generate text that is not only coherent but also contextually relevant. In our project this means generating cohesive and creative poems.

The model was trained over 5 epochs. An epoch represents a complete pass over the entire training dataset. Running several epochs is crucial as it allows the model to iteratively learn from the data, adjusting its parameters each time to better predict the next word in a sequence. 5 epochs were chosen as the model was showing signs of over fitting at the fifth epoch.

A batch size of 2 was used, meaning that the model updates its parameters after every two samples of data. This relatively small batch size helps in managing memory usage and can provide a more robust gradient estimate at each step, which is particularly useful for complex models like GPT-2 that can easily overfit to training data.

The learning rate is a key hyperparameter that controls how much to change the model's parameters in response to the estimated error each time the model weights are updated.

Choosing the right learning rate is essential for good training performance; too small a rate can lead to a long training process, and too large a rate can cause the training to diverge. We chose the default learning rate of 5e-4 for the model training as it appeared to fit the model and performed well in training.



```
epochs = 5
learning_rate = 5e-4
warmup_steps = 1e2
epsilon = 1e-8

# this produces sample output every 100 steps
sample_every = 100
```

The cross-entropy loss function was employed, which is standard for training language models. This loss function measures the model's performance by calculating the difference between the predicted probabilities and the actual distribution of the words.

$$L = -\sum_{i=1}^{n} y_i log(p_i)$$

Here ‘n’ is the number of classes, ‘Yi’ is the true probability distribution, ‘Pi’ is the predicted probability distribution.

Here's how it fits into the training:

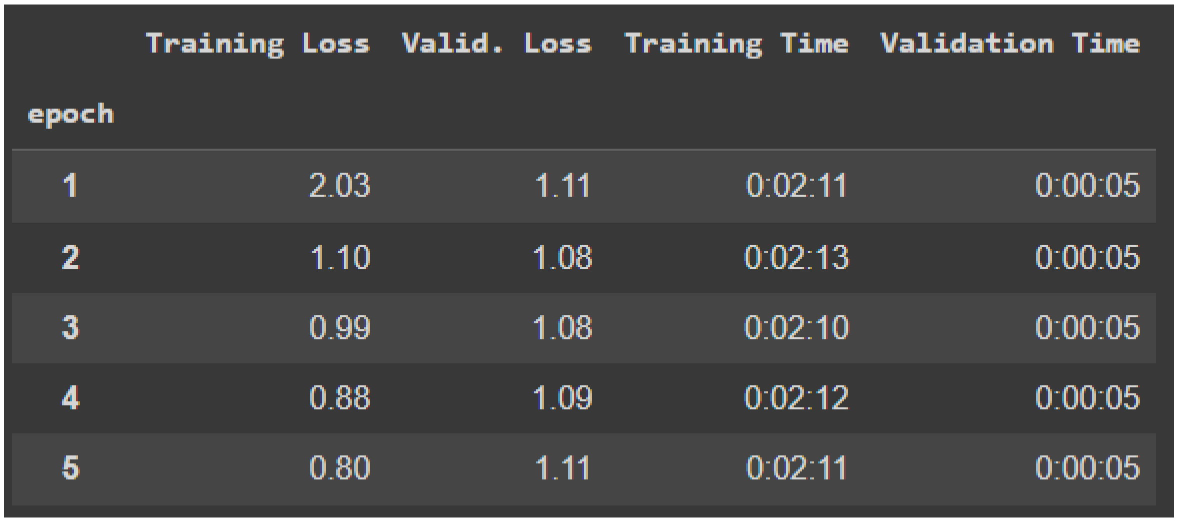
Quantifying Error: Cross-entropy provides a measure of dissimilarity. It quantifies how different the predicted probability distribution for the next word is from the actual distribution (the true next word).
Gradient Calculation: During backpropagation, the gradient of the cross-entropy loss is calculated to update the weights. These gradients indicate the direction in which the weights should be adjusted to minimize the error.
Model Optimization: By minimizing the cross-entropy loss over several iterations, the model learns to assign higher probabilities to more likely next words, enhancing its predictive accuracy and the quality of the generated text.

The training loop involved feeding batches of tokenized text to the model, calculating the loss for each batch, and updating the model's weights using an optimizer (typically Adam or similar).

The use of a validation set helped monitor the model's performance and avoid overfitting by providing feedback on how well the model generalizes to new, unseen data.

4. Results

Training and Validation Loss



epoch	Training Loss	Valid. Loss	Training Time	Validation Time
1	2.03	1.11	0:02:11	0:00:05
2	1.10	1.08	0:02:13	0:00:05
3	0.99	1.08	0:02:10	0:00:05
4	0.88	1.09	0:02:12	0:00:05
5	0.80	1.11	0:02:11	0:00:05

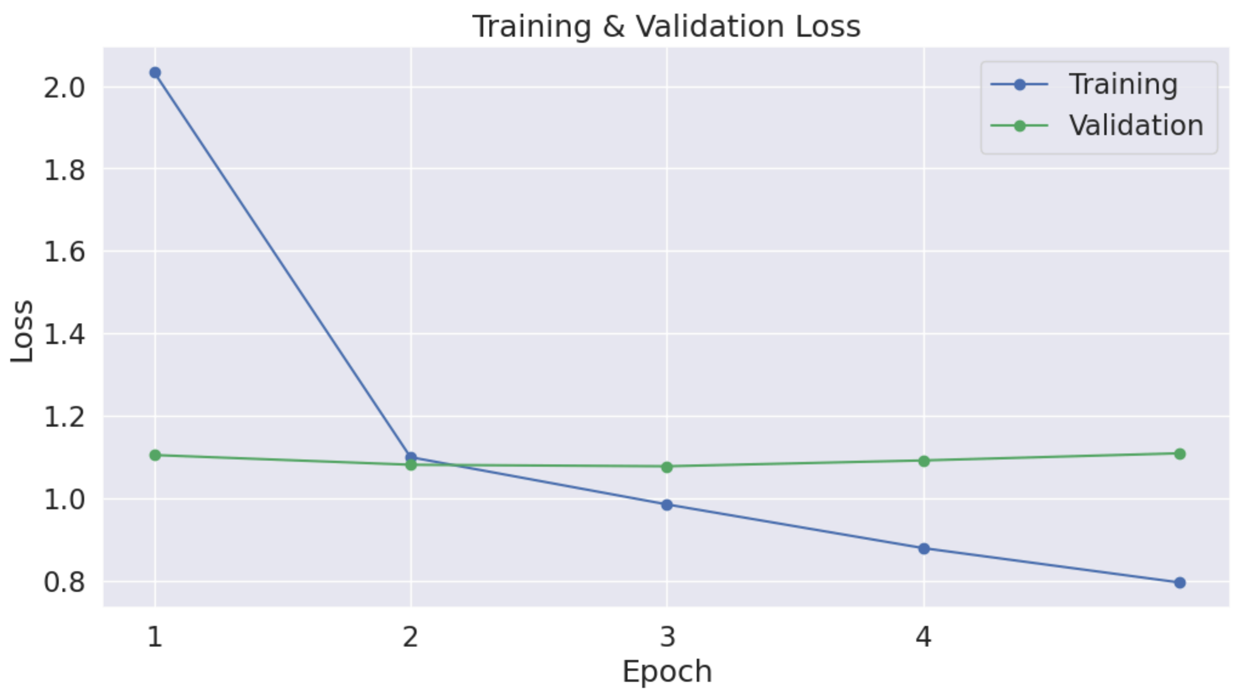
The training and validation loss trends observed over five epochs provide key insights into the model's learning dynamics. From the epoch data:

Training Loss: The training loss consistently decreases from 2.03 in epoch 1 to 0.80 in epoch 5. This indicates that the model is effectively learning from the training dataset, optimizing its weights and biases to minimize the error in its predictions.

Validation Loss: The validation loss initially decreases, reaching its lowest at 1.08 in epochs 2 and 3, but then slightly increases, indicating a potential overfitting issue as the model trains further. By the final epoch, the validation loss has increased to 1.11.

These observations suggest that while the model is becoming better at fitting the training data, it might be starting to memorize the training data too closely, reducing its ability to generalize to new, unseen data. This is hinted by the uptick in validation loss in the later epochs, despite ongoing improvements in training loss.

The line graph vividly illustrates the trends mentioned above. The training loss (blue line) shows a steady decline, demonstrating improving model performance on the training set across epochs. In contrast, the validation loss (green line) plateaus and slightly increases after the third epoch, indicating that the gains in training performance are not translating as effectively to unseen data.



This pattern suggests a few potential issues and considerations:

Early Stopping: Implementing early stopping could prevent overfitting by halting training when validation loss begins to increase, potentially after the third epoch in this case.

Model Complexity: The model might be too complex, capturing noise and details specific to the training data that do not generalize well.

Regularization Techniques: Applying techniques such as dropout, L1 or L2 regularization might help in reducing overfitting by penalizing overly complex model weights.

Samples of poems generated by the final model:

A hand that raises his wand,
The wand is like a glove,
A lamp like the wind;
But when in the window, at noon
I see him shining,
The other hand raises his wand:
And when my husband prays
To make me come home,
I keep the wand in my hand:

And when he is done, I put his head in the dust;
He lies here with me still,
And the wind beats on him, till I drown.
The day is now past, when I came,
And all is old and empty.

And this is what the elders call me,
A bride, and this is what the elders call me.

And since the elders say that it is better for me to be alone,
I have got hold of this wand.

==

I went to the door
And the keeper of the hall gave me a handkerchief
To hold and record the evening of my stay
As all the world heard the stories of the great house-fires.
In that hall the master kept a secret,
And to my own astonishment I saw
His black marble steps branching along the crystal street
And slowly raising their ominous alarms.
But now the old master has left me
And I am only shadow of him.

==

Let them be your guide:
But no more they will say
Thou shalt not answer.
For these eyes thou, Stella'st, are not blind,
Nor in vain strive to see;
Nor in the morning shall they bid farewell

==

Model strength and limitations Based on generated poems analysis

Strengths:

- Thematic Consistency: The model shows a strong ability to maintain thematic integrity across different poems.
- Imagery and Symbolism: There is a rich use of imagery and symbolism that enhances the poetic quality and depth.
- Complex Language Handling: The model adeptly manages complex language structures, maintaining fluidity and coherence in the poems.

Limitations:

- Emotional Depth Variation: While some poems exhibit emotional insights, others might benefit from more varied emotional tones to better reflect the complexities of the themes addressed.
- Narrative Clarity: Some poems might appear slightly disjointed or lacking in a clear narrative thread, which could be improved by enhancing the model's contextual understanding.

While the poems generated may seem absurd and the un-understandable, the above analysis is done with the mindset of the model being not as performant as a creative human and the analysis is not based on emotionality as often it is with human generated poems.

5. Evaluation

To evaluate the performance of our GPT-2 model to generate poems, we have chosen two methods of evaluation:

a. Manual evaluation
b. GPT2 model based loss evaluation

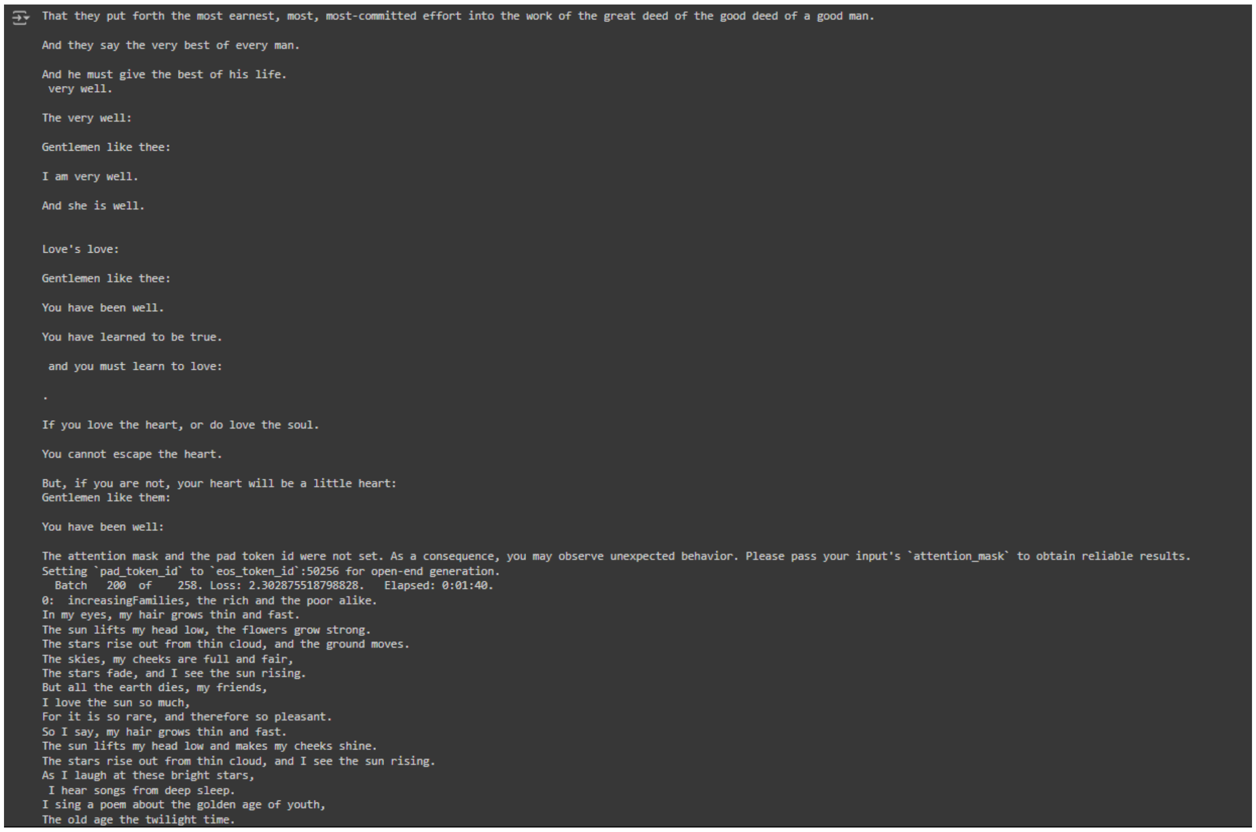
a. Manual Evaluation:

Here, we assess the quality of the poem which is generated by our GPT-2 model by some metrics such as creativity, coherence, and impact on the reader.

Let us look into some of the poems (epochs) that are generated during the training of the GPT-2 model for evaluating the performance manually:

Poem generated - 1:

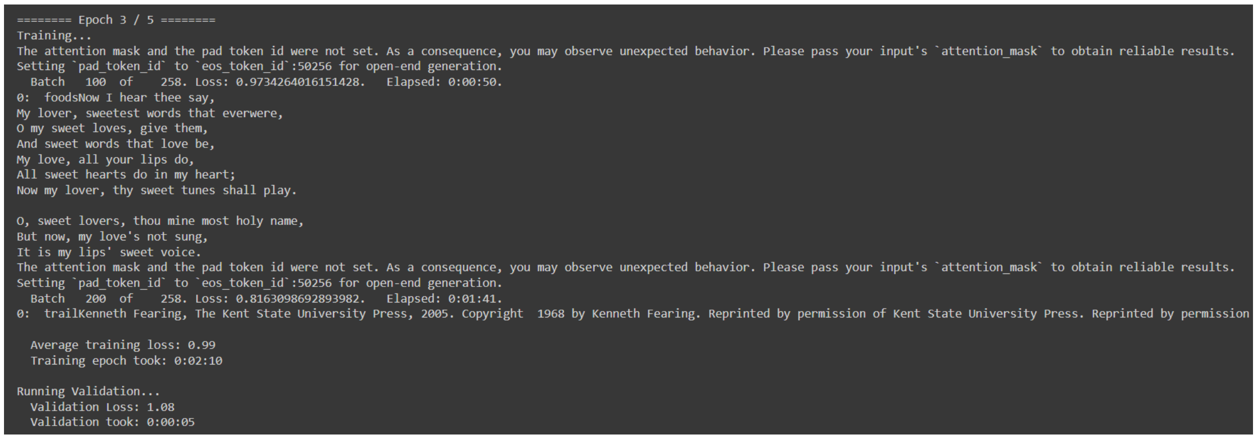
Here, as this is the first poem that was generated, as we can see the structure of the poem is well maintained and sometimes there is ‘:’ symbol at every end of the sentence and also the coherence is good but not excellent as we required the poem to be. The poem lacks a bit of creativity as we can see the rhyming is made with the same word ‘well’ and the impact is also a bit less.



```
That they put forth the most earnest, most, most-committed effort into the work of the great deed of the good deed of a good man.

And they say the very best of every man.

And he must give the best of his life.
 very well.

The very well:

Gentlemen like thee:

I am very well.

And she is well.


Love's love:

Gentlemen like thee:

You have been well.

You have learned to be true.

 and you must learn to love:

.

If you love the heart, or do love the soul.

You cannot escape the heart.

But, if you are not, your heart will be a little heart:
Gentlemen like them:

You have been well:

The attention mask and the pad token id were not set. As a consequence, you may observe unexpected behavior. Please pass your input's `attention_mask` to obtain reliable results.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
  Batch   200  of    258. Loss: 2.302875518798828.   Elapsed: 0:01:40.
0:  increasingFamilies, the rich and the poor alike.
In my eyes, my hair grows thin and fast.
The sun lifts my head low, the flowers grow strong.
The stars rise out from thin cloud, and the ground moves.
The skies, my cheeks are full and fair,
The stars fade, and I see the sun rising.
But all the earth dies, my friends,
I love the sun so much,
For it is so rare, and therefore so pleasant.
So I say, my hair grows thin and fast.
The sun lifts my head low and makes my cheeks shine.
The stars rise out from thin cloud, and I see the sun rising.
As I laugh at these bright stars,
 I hear songs from deep sleep.
I sing a poem about the golden age of youth,
The old age the twilight time.
```

Poem generated - 3:

Here, as this is the third poem that was generated by our model, as we can see the structure of the poem improved a lot by making it more compact and added more meaning to it, and also the coherence is very good and makes the poem as if it was following a theme or a narrative style. The poem is a bit more creative when compared to the poem generated before, as the rhyming words changed and also new synonyms of words which sound very coherent with the entire poem theme.



```
======== Epoch 3 / 5 ========
Training...
The attention mask and the pad token id were not set. As a consequence, you may observe unexpected behavior. Please pass your input's `attention_mask` to obtain reliable results.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
  Batch   100  of    258. Loss: 0.9734264016151428.   Elapsed: 0:00:50.
0:  foodsNow I hear thee say,
My lover, sweetest words that everwere,
O my sweet loves, give them,
And sweet words that love be,
My love, all your lips do,
All sweet hearts do in my heart;
Now my lover, thy sweet tunes shall play.

O, sweet lovers, thou mine most holy name,
But now, my love's not sung,
It is my lips' sweet voice.
The attention mask and the pad token id were not set. As a consequence, you may observe unexpected behavior. Please pass your input's `attention_mask` to obtain reliable results.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
  Batch   200  of    258. Loss: 0.8163098692893982.   Elapsed: 0:01:41.
0:  trailKenneth Fearing, The Kent State University Press, 2005. Copyright  1968 by Kenneth Fearing. Reprinted by permission of Kent State University Press. Reprinted by permission

  Average training loss: 0.99
  Training epoch took: 0:02:10

Running Validation...
  Validation Loss: 1.08
  Validation took: 0:00:05
```

Hence, with the help of the metrics like coherence, creativity, emotional impact, and sometimes the poetic language plays a major role, as here we have not used it we went with the first three mentioned metrics to determine whether the poem generated by the GPT-2 model is consistent or not, then to finally evaluate the model if the performance is diminishing or improving

b. GPT-2 model based loss evaluation:

The model uses a variant of cross-entropy loss, specifically known as the negative log-likelihood loss during training. It looks for any dissimilarity between the probability distribution of the generated poem or the predicted distribution over the entire corpus and also the true target distribution (the ground truth).

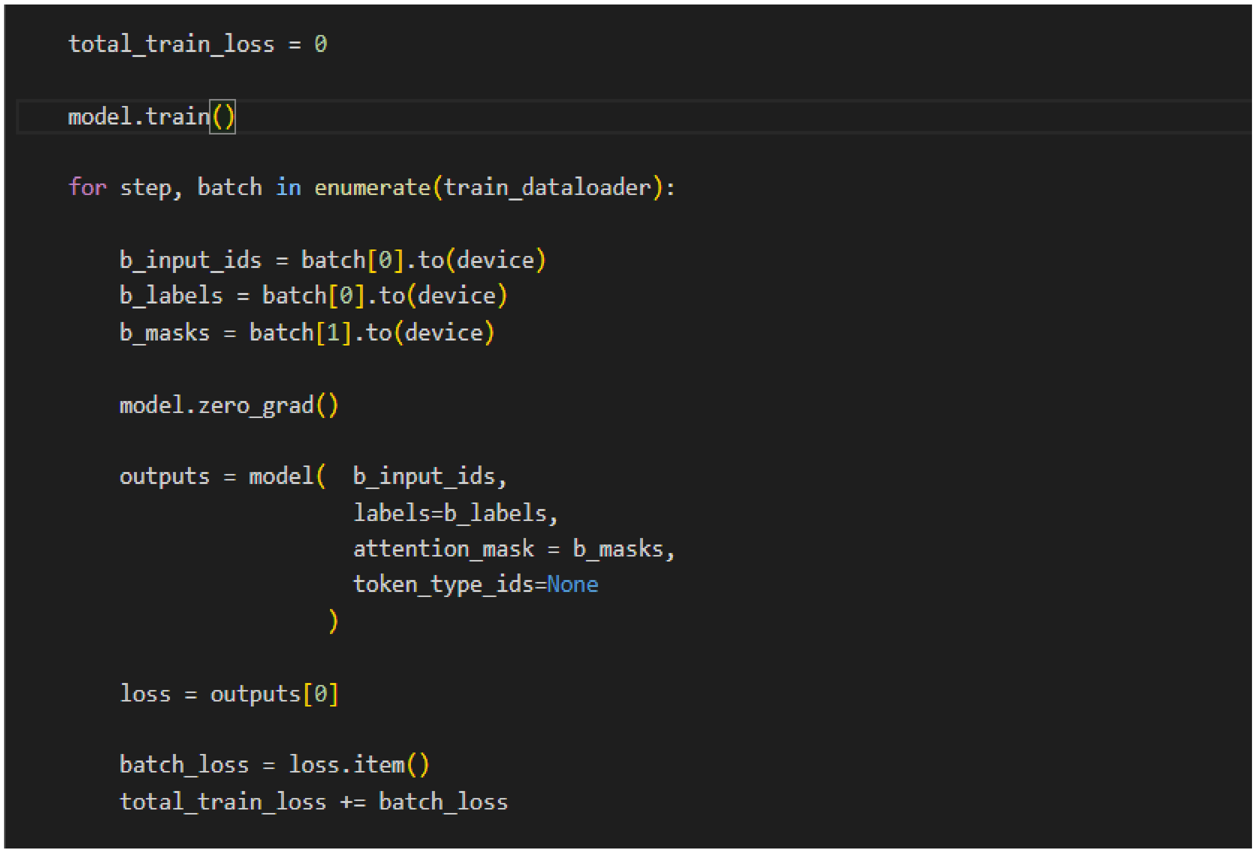
The calculation for this loss is:

$$\sum_{i=1}^{n} y_i \log(p_i)$$

- n is the number of classes
- Yi is the true probability distribution
- Pi is the predicted probability distribution

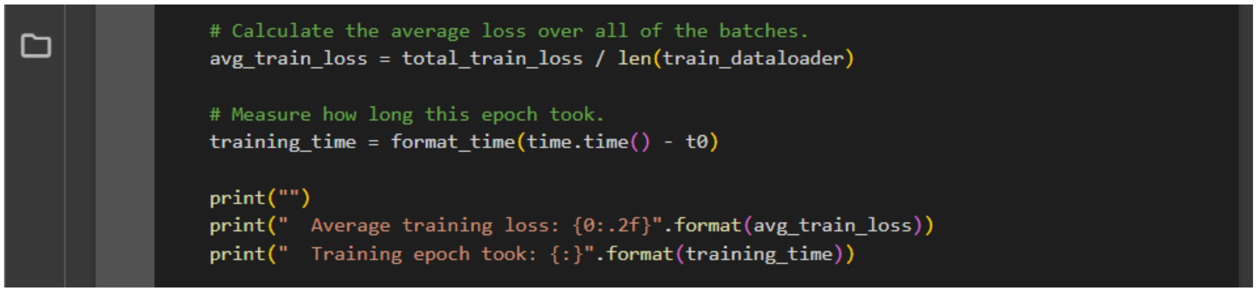
Here the main motive to train the model is to make this loss minimum as less as possible, the lower the value is the better the performance of the model.

In the below code, we are calculating the loss of each generated poem and adding them to the total train loss to determine the average loss, which is the loss of the GPT-2 model.



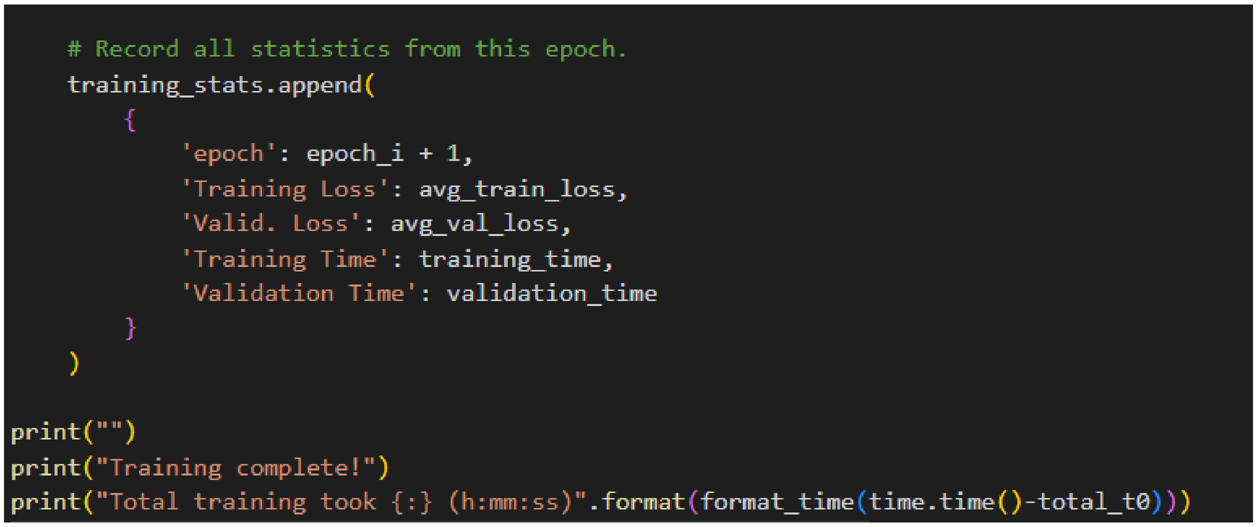
```
total_train_loss = 0

model.train()

for step, batch in enumerate(train_dataloader):

    b_input_ids = batch[0].to(device)
    b_labels = batch[0].to(device)
    b_masks = batch[1].to(device)

    model.zero_grad()

    outputs = model(  b_input_ids,
                      labels=b_labels,
                      attention_mask = b_masks,
                      token_type_ids=None
                    )

    loss = outputs[0]

    batch_loss = loss.item()
    total_train_loss += batch_loss
```

Total batch loss → average loss is generated



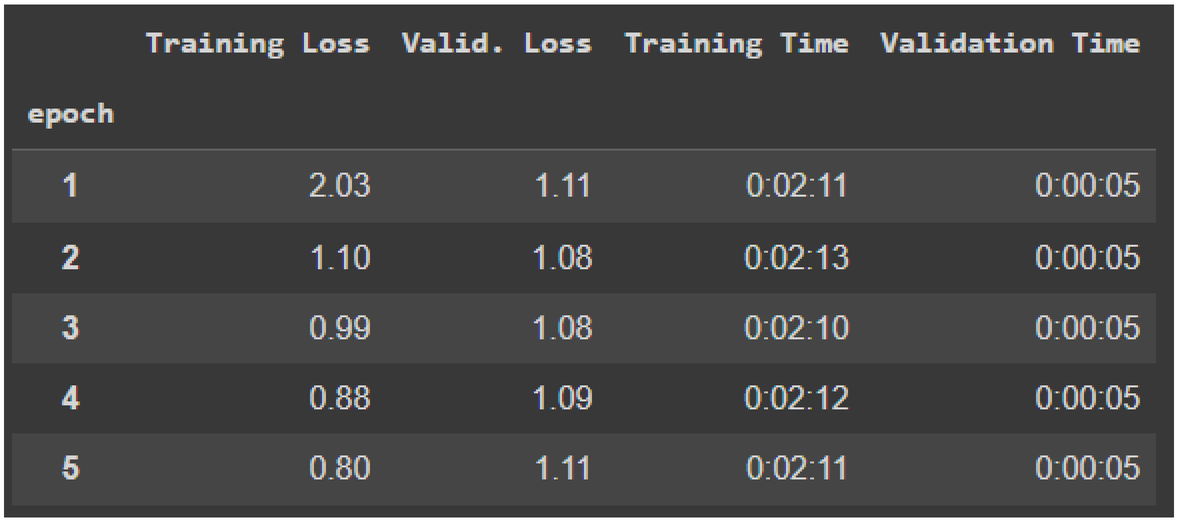
```
# Calculate the average loss over all of the batches.
avg_train_loss = total_train_loss / len(train_dataloader)

# Measure how long this epoch took.
training_time = format_time(time.time() - t0)

print("")
print("  Average training loss: {0:.2f}".format(avg_train_loss))
print("  Training epoch took: {:}".format(training_time))
```

Recording all statistics from each epoch i.e. from each poem generated including the time taken for each epoch to run while training.



```
    # Record all statistics from this epoch.
    training_stats.append(
        {
            'epoch': epoch_i + 1,
            'Training Loss': avg_train_loss,
            'Valid. Loss': avg_val_loss,
            'Training Time': training_time,
            'Validation Time': validation_time
        }
    )

print("")
print("Training complete!")
print("Total training took {:} (h:mm:ss)".format(format_time(time.time()-total_t0)))
```

Here is the table which shows the evaluation based on the GPT2 based model's loss function for each poem that has been generated.



epoch	Training Loss	Valid. Loss	Training Time	Validation Time
1	2.03	1.11	0:02:11	0:00:05
2	1.10	1.08	0:02:13	0:00:05
3	0.99	1.08	0:02:10	0:00:05
4	0.88	1.09	0:02:12	0:00:05
5	0.80	1.11	0:02:11	0:00:05

As it can be seen, the training loss has been consistently reduced, which is a sign that our GPT2 model is getting better and better at training for epoch it ran on training data, but the model started overfitting at 5th epoch, as the validation loss has been increased slightly, hence we stopped training the model after 5 epochs and concluded this evaluation of the GPT2 model's performance to generate poems.

6. Conclusion

The exploration into the realm of automated poetry through the utilization of the GPT-2 model has produced significant insights and results, marking a notable advancement in the use of artificial intelligence in creative writing. Our project has demonstrated that a pre-trained language model, when finely tuned with a specialized dataset, can effectively mimic the stylistic and thematic nuances of human poets, producing poetry that resonates with thematic depth and linguistic complexity.

Throughout the course of this study, several key outcomes were observed. The GPT-2 model showcased a robust ability to adapt to the poetic styles present in the training dataset. This was evidenced by the decreasing training loss and the qualitative attributes of the generated poems, which often mirrored the complexity and imagery typical of human-crafted poetry, if not the cohesiveness or depth.

While the model performed well on the training set, the slight increase in validation loss during later epochs signaled the beginning of overfitting. This highlights the challenge in machine learning of balancing model complexity with the ability to generalize to new data.

This project contributes to the broader discourse on the role of AI in creative fields. By generating new poetic compositions, the model not only serves as a tool for artistic creation but also prompts a deeper reflection on the nature of creativity and the potential for machines to engage in traditionally human-centric domains.

In conclusion, the project has not only achieved its goal of generating poetry through the capabilities of AI but has also opened up new avenues for understanding and advancing the synergy between technology and human creativity. It stands as an example of the potential of AI to become a valuable tool in the realm of artistic expression, providing the ability to augment, rather than replace, the creative capacities of humans.