**CSCE 5290- Natural Language Processing**

**Final Project**

**Text Summarization Using Pre-Trained Language Models by Fine-Tuning.**

Brijesh Rao Pamara

Rohith Reddy Bollareddy

Amrutha Veeramachaneni

Nikhil Reddy Vemireddy

**Introduction:**

GPT-2 (Generative Pre-trained Transformer 2) is a state of art transformer model created by Open AI which implements a decoder-only transformer architecture based neural network that uses attention rather than recurrent and convolutional based models. This mechanism allows the model to focus on most relatively significant parts which allows increased parallelization rather than other models.

The main purpose of this project is to summarize the overall content of the text to a meaningful text which is relatively smaller than the original text but preserves its overall theme and meaning. Before the advent of deep learning, the approach to Natural Language processing was mostly rule based with simple Machine Learning algorithms. But, today, with deep learning and topics of Natural language processing enabled models to be trained and analyze the text in a non sequential manner increased greatly.

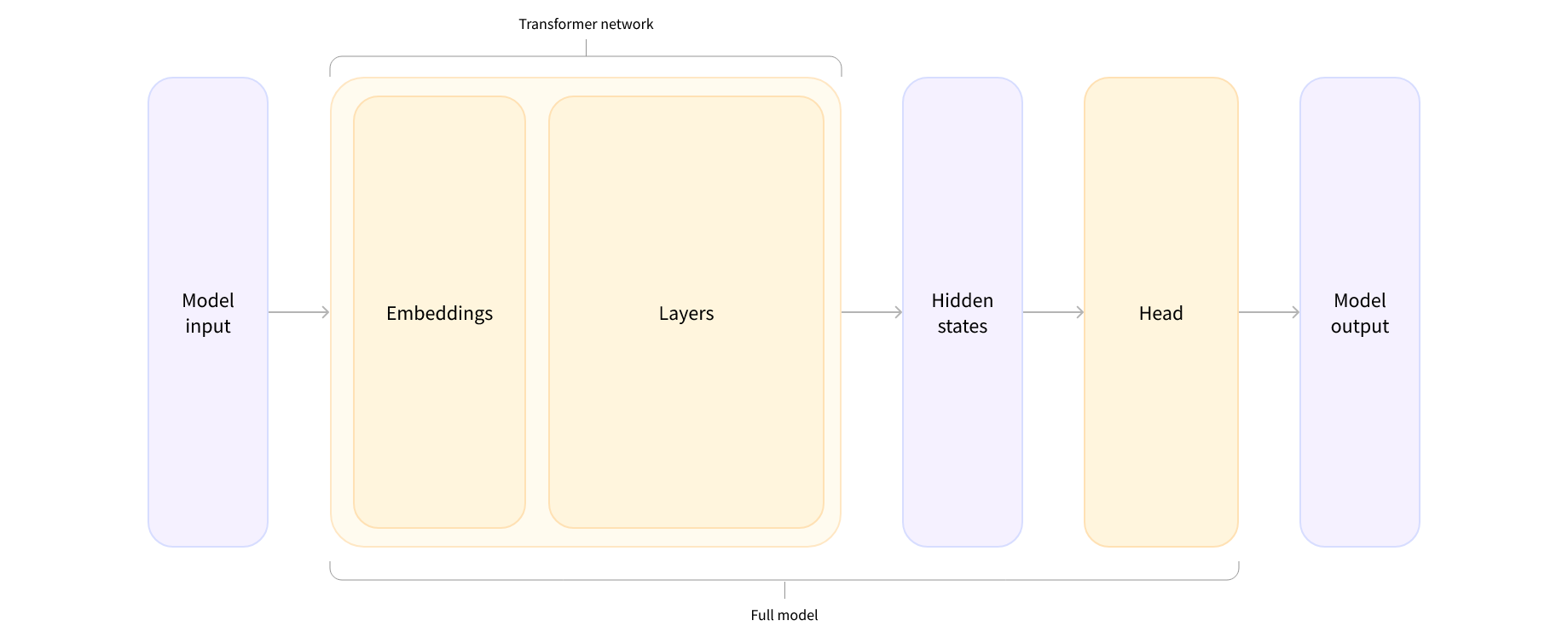
The attention mechanism that is used in GPT-2 proved to be extremely rewardful in drawing the generalized context of the text. In today’s world, a lot of business entities take the reviews and surveys of their products and services which enables them to develop and modify according to the market and in customer interest. Text summarization comes in use to draw important and valid information.

**Background:**

Pre-trained Language Models (LMs) has shown good results in terms of performance and efficiency in tasks like text classification, text summarization, POS tagging etc.., it is not clear as to how pre-trained LMs can be used for text generation like abstractive summarization in terms of improving the sample efficiency. Prior work has shown experiments using loading pre-trained weights into a decoder network where the transformer language model encodes the text and creates the summary. This further takes into account that all the attributes in the language model are pre-trained before it goes to fine-tuning. Experiments also show that pre-trained transformer Language models with fine tuning on a specific dataset gives better results when compared to pre-trained transformed encoder-decoder networks.

**Model and Workflow:**

The below diagram shows the architecture of the transformer model, specifically the GPT2 model as these models only contain the decoder part, as they are a decoder only architecture of transformer models. The model input is first converted into the sentence embeddings and the tokens are passed onto the attention layers, where the attention mechanism attends to each token of the input text and computes the similarity of all the tokens with respect to each other. The outputs are then fed into a hidden layer consisting of multi-layer perceptrons which act as a feed forward layer and these then generate an output prediction.

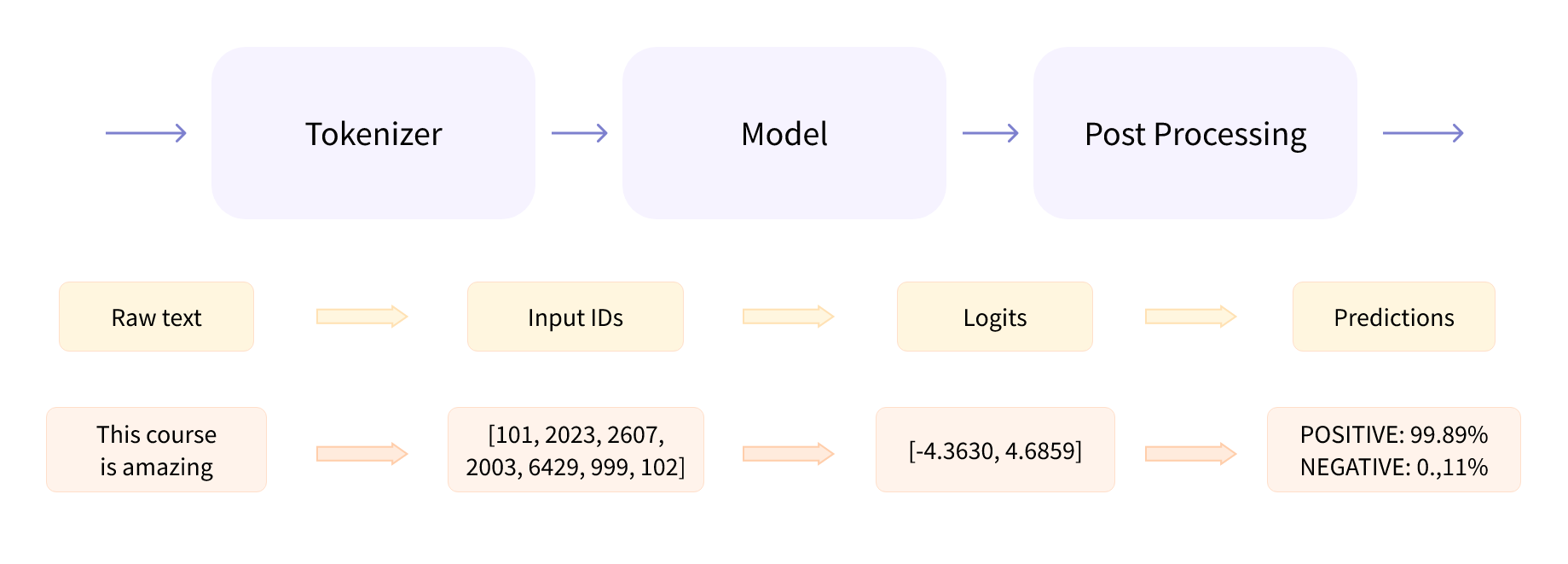


GPT model architecture

The project work flow is as follows:

The Raw text is first tokenized using the GPT2 tokenizer. The tokenized text is then fed into a pre-trained model which we finetune using our dataset of reviews. Then additional post-processing is applied on the generated text which is used to clean the outputted text.

Then the generated text is compared against the expected output. The training loss is computed on whether the expected output and the generated output are similar as defined by a loss function. In our case we are using a cross-entropy loss function.



Project architecture

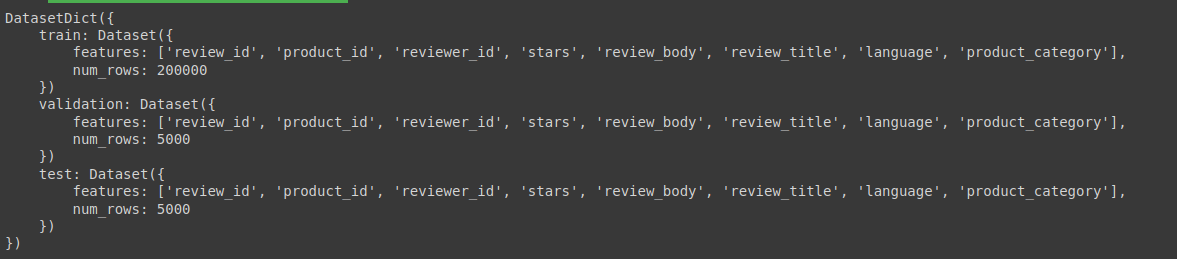
**Dataset:**

The dataset is the amazon reviews corpus it consists of a multilingual dataset of reviews of amazon product reviews. We only use the English language data of the corpus. It consists of a training set of 20000 reviews and testing and validation sets of 5000 reviews each.

Each entry consists of 'review\_id', 'product\_id', 'reviewer\_id', 'stars', 'review\_body', 'review\_title', 'language', 'product\_category'.

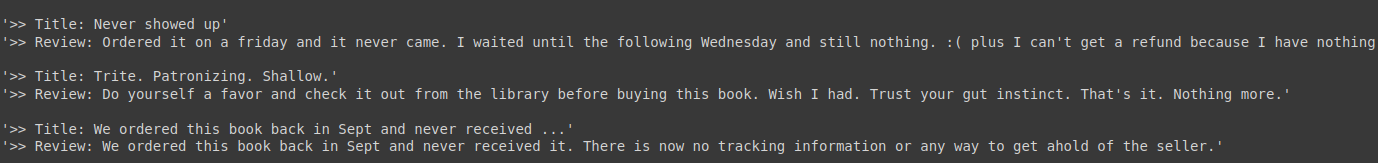
Link to the dataset repository: <https://registry.opendata.aws/amazon-reviews-ml/>.

In this project we are only focused on the ‘review\_body’ and ‘review\_title’ columns of the dataset, as these are the fields which contain the text and title, which we use as the summary.



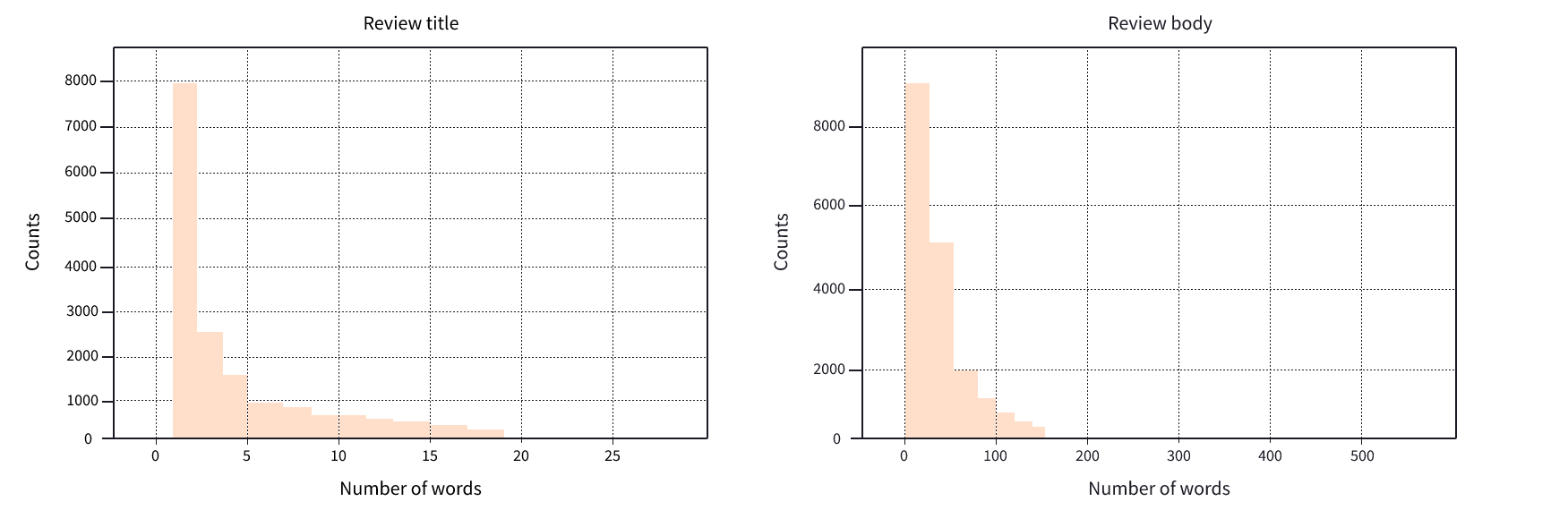
Summary of the dataset

Sample text from the repository:



**Analysis:**

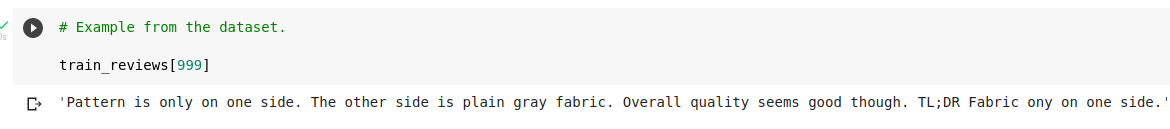
As part of preprocessing the database, we perform basic analysis operations on the dataset, like cleaning the text of any special characters. We also analyze the distribution of the dataset to see if there is any imbalance of data. In the dataset we find that a significant amount of the summary in the product review are one or two words, if we train the model on this imbalance it will then we will have a clear bias in the output. Hence, we filter our dataset to remove any summaries which are less than two words. So that we can get higher quality summaries.



**Implementation:**

The main parts of the project implementation are loading the dataset and cleaning it to make a normalized form where no feature or attribute are over-represented to avoid overfitting along that particular feature as explained in the sections above.

As GPT2 is a decoder only architecture both the summary and the review body needs to be input in the model along a single sentence, this necessitates a special token which demarcates the review body from the summary. We use the word " TL;DR " as this special token. We then tokenize this word with the gpt2 tokenizer so it has a unique id. We then join the review body and summary into a single sentence.



Example of a sentence with review body and summary.

Then we take the processed dataset and need to tokenize all the sentences in the dataset, which we implement using the GPT2 tokenizer. We define a pytorch dataloader which is needed to input our tokenized data into the model during training. The data loader converts the tokenized sentences into torch tensors which are used by pytorch framework to process the data in the model, and compute the forward and backward passes.

We then import a pre-trained GPT2 language model from the transformers library as the baseline model. This model forms our baseline text summarizer and is compared against the fine tuned model trained on the amazon reviews dataset to measure how the output differs based on the task of fine tuning.

The GPT2 model consists of the attention mechanism followed by a multi layer perceptron. Hence, the attention mechanism forms the backbone of our fine tuning process.

Here is a simplified example of how the attention mechanism works, using pseudo code:

# Input sequence of elements x1, x2, ..., xn

# Query vector q

# Calculate similarity between query and each element

attention\_weights = []

for i in 1 to n:

attention\_weights[i] = similarity(q, x[i])

# Normalize attention weights to sum to 1

attention\_weights = attention\_weights / sum(attention\_weights)

# Calculate weighted sum of input elements

output = sum(attention\_weights[i] \* x[i] for i in 1 to n)

In this code, the attention mechanism first calculates the similarity between the query vector and each element in the input sequence. These similarities are then used to compute attention weights, which indicate the relative importance of each element in the input. The attention weights are then normalized to sum to 1, and the weighted sum of the input elements is calculated and used as the output of the attention mechanism.

This output is then used as the input to the next layer in the transformer model, where it is processed further and used to generate a prediction or output. The attention mechanism can be applied multiple times in a transformer model, allowing the model to attend to different parts of the input sequence at different stages of processing. This helps the model to capture complex dependencies and relationships between the input elements, and can improve its performance on various natural language processing tasks.

We then feed our tensored data as input to the model during the training face where the similarity between all the tokens are calculated and the loss is calculated. We use the cross-entropy loss between the review-body and summary to finetune the model during the training phase. The Adam optimizer, a variant of the Stochastic Gradient descent algorithm, is used in the backpropagation phase to update the weights of the network. We define a batch size of 32 and train for 1 epoch, as anymore than that it seems to overfit the model to the training data and we could observe no decrease in the model loss.

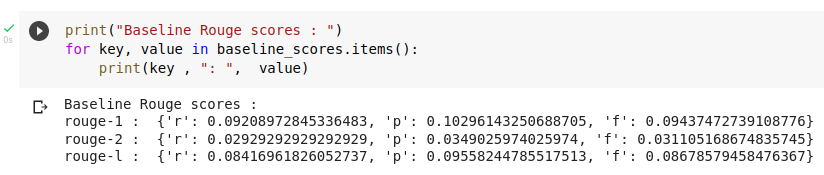
We then define an inference step where we use the model to generate the tokens for a given input which represent the output summary generated by the model. The output tokens are then assembled and decoded to form the sentence.

Finally, we use both the baseline model and the fine tuned model to generate sample summaries and test them against the actual summaries found in the test data, which is then used for evaluation.

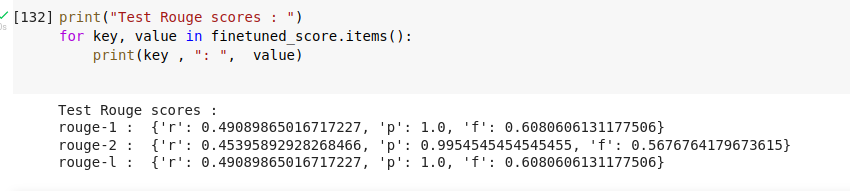
**Results:**

**ROUGE** score abbreviated for **Recall-Oriented Understudy for Gisting Evaluation**, is a score used for evaluating automatic summarization of text in natural language processing. The metrics compare a summary produced by a model, against a reference summary, to evaluate its accuracy.

Accuracy score with baseline model:



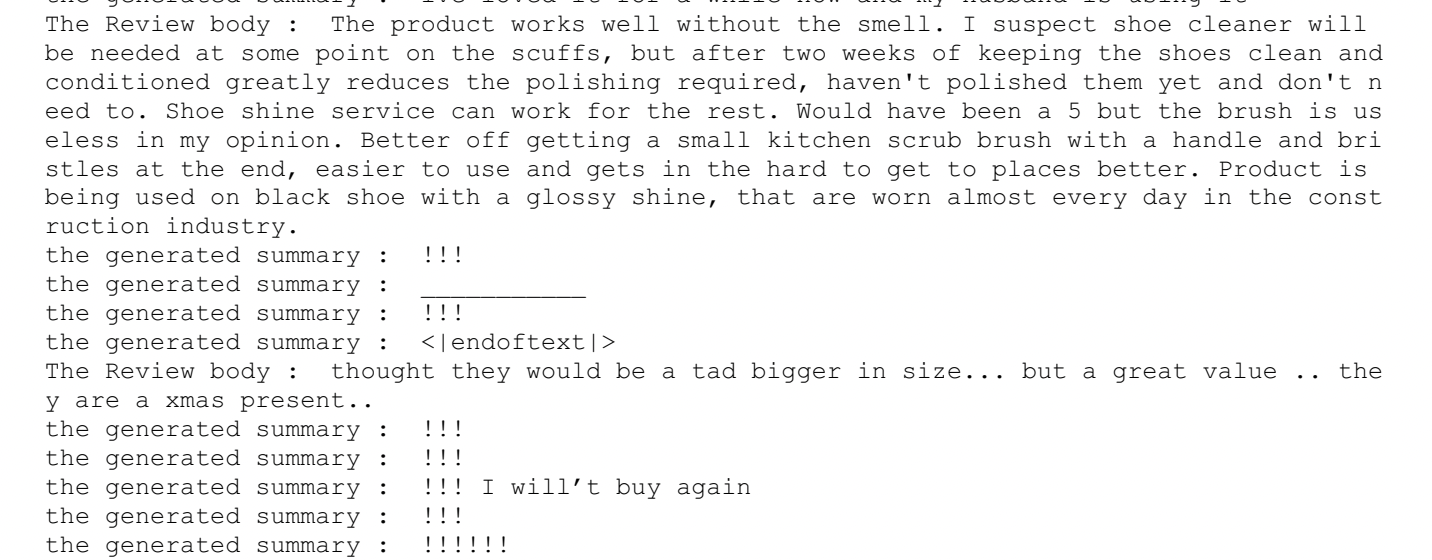
Accuracy score with a pre-trained GPT model with fine tuning on the dataset:



From the above two images we can see the evaluation scores of our baseline GPT model with no finetuning and the GPT2 model which has its weights fine tuned by training on our dataset. We can see the scores have dramatically improved and that the summaries provided by the model improve dramatically when compared to the prior baseline model.

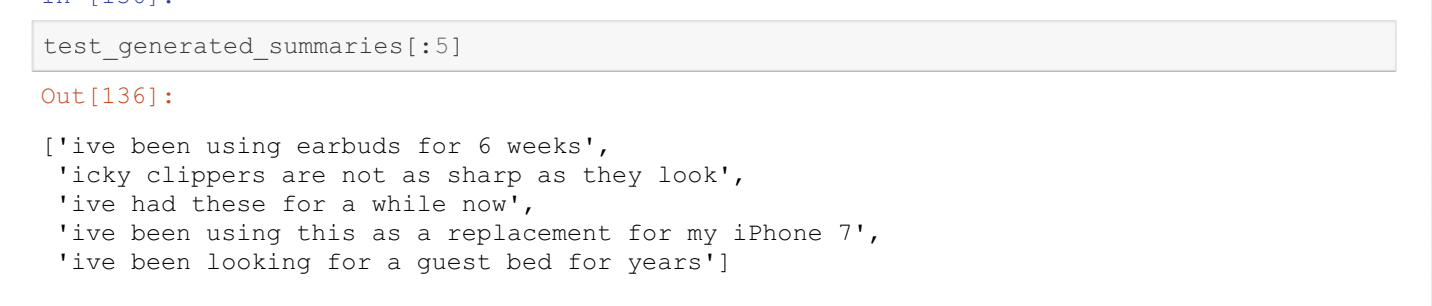
Baseline Summary:

An example of the text generated from the baseline model we notice that the reviews are highly variable and oftentimes contains special characters.



Fine tuned Summary:

An example of summaries generated from the fine tuned dataset, the generated text is relevant and has a good grammatical structure.



**Project Management:**

Examined the dataset and its properties cleaned up and transformed the data into a form suitable for model creation.

Completed the preliminary data analysis and normalized the dataset by eliminating entries causing imbalance in the dataset to minimize the bias in the dataset, which could impact model performance.

Created a baseline model, which is the pre-trained GPT- 2 model and we compared this model’s performance against the fine tuned GPT- 2 model which we have trained on our amazon reviews dataset.

Imported and built a pre-trained GPT model from the model weights and evaluated its performance with samples from the dataset to generate summaries.

Evaluated the baseline and GPT model using the ROUGE score, used to evaluate summarization.

**Responsibility:**

| Dataset Loading, Cleaning of dataset and data preprocessing | Rohith Reddy, Brijesh Rao |
| --- | --- |
| Creating Baseline model,data loaders, tokenizing the text to feed into model | Amrutha, Nikhil Reddy |
| Defining the GPT-2 model and training the model and Generating sample summaries | Brijesh, Amrutha |
| Generating baseline Summary and test summary, Evaluation of baseline and fine tuned models and Generating Rogue Score | Nikhil, Rohith |
| Documentation | Brijesh, Rohith, Amrutha, Nikhil |

**Contribution:**

Brijesh Rao Pamara- 25%

Rohith Reddy Bollareddy- 25%

Amrutha Veeramachaneni- 25%

Nikhil Reddy Vemireddy- 25%

**References:**

1. [Sample Efficient Text Summarization Using a Single Pre-Trained Transformer](https://arxiv.org/abs/1905.08836)
2. [Language Models are Unsupervised Multitask Learners](https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdfx)
3. [The amazon reviews corpus](https://registry.opendata.aws/amazon-reviews-ml/)
4. [Hugging Face Transformers](https://github.com/huggingface/transformers)

**GitHub Repo:**

<https://github.com/rohithbollareddy/CSCE-5290-natural_language_processing>