Classifying Large Language Models based on their Text Generation

Balaji Udayagiri, Yashita Watchpillai

Abstract

This project classifies Large Language Models based on their text completions of set of truncated sentences. In this study we classify the text completions of 5 LLMs in total GPT-2, GPT-Neo, Facebook OPT, Gemma2:2B and Mistral-OpenOrca. A Fully Connected (FC) neural network classifier is trained on SBERT embeddings of generated texts. . The project offers insights into the capabilities of different LLMs in completing political discourse and demonstrates the efficacy of SBERT-based classification in distinguishing outputs from various models. We evaluate the Classification model on the parameters of precision, recall, and F1-score. Our model achieves an accuracy of 70.83%.

1 Introduction

Different AI models, such as GPT-2, GPT-Neo, or specialized models like Gemma and Mistral, have unique architectures and training methodologies that influence their output style and content. Understanding which model generated a specific text can provide insights into its stylistic and thematic features. For example, the ability to distinguish between generative models could help researchers and practitioners recognize patterns in creativity and language usage that are characteristic of certain architecture.

In applications where AI-generated content is used—such as journalism, marketing, and content creation—being able to identify the source model can enhance accountability. Different models may have different levels of reliability, bias, or accuracy in their outputs. Identifying the model can help users assess the credibility and appropriateness of the content, especially in sensitive domains like politics or healthcare.

Analyzing the outputs of different models can provide valuable feedback for developers and researchers. By understanding how different models perform in generating specific types of content, they can make more informed decisions about model training, fine-tuning, and the selection of models for particular applications .

The utilization of Large Language Models (LLMs) has grown significantly in natural language processing (NLP) tasks such as text generation, translation, and summarization. Pre-trained models, including GPT-2, GPT-Neo, and Facebook's OPT from Hugging Face, as well as Gemma2:2B and Mistral-OpenOrca from Ollama, offer scalable solutions for generating human-like text. We want to understand the underlying difference in the generated texts of these models. We further analyze their performance using a classification task powered by Sentence-BERT (SBERT) embeddings to distinguish model-specific output characteristics.

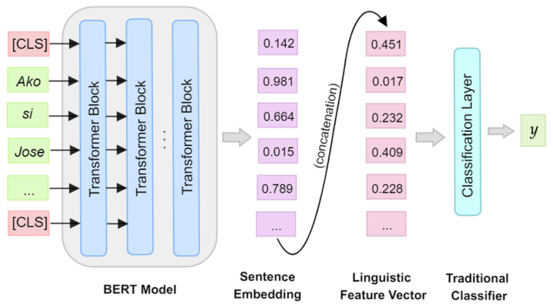

Figure 1: S-Bert Architecture

2 Literature Review

In recent years, Large Language Models (LLMs) have become central to numerous Natural Language Processing (NLP) tasks, including text generation, classification, and completion. These models, such as GPT, BERT, and their variants, have demonstrated a remarkable ability to generate human-like text and solve a variety of NLP tasks. This literature review explores some of the key papers that lay the foundation for understanding and comparing the capabilities of different LLMs, particularly for tasks like identifying the model used in sentence completion, which is the focus of the current project.

The introduction of GPT-3 by Brown et al.in [1] ”Language Models are Few-Shot Learners” marked a significant leap in NLP capabilities. GPT-3 demonstrated that with enough parameters, an LLM could perform tasks with minimal fine-tuning or task-specific training data. This paper is highly relevant as it discusses how models generate text given a prompt, which aligns with the sentence-completion aspect of the current task. The ability of GPT-3 to handle few-shot learning implies that it may generate more varied and complex sentence completions, making it one of the more challenging models to classify based on its output.

Devlin et al. introduced BERT in[3] (Bidirectional Encoder Representations from Transformers) as a transformer-based model pre-trained on large corpora, emphasizing the importance of context in both directions (left-to-right and right-to-left). Unlike models like GPT that are unidirectional, BERT's bidirectional nature allows for a deeper understanding of the text, which can influence how it completes sentences. BERT is widely used in NLP tasks requiring context awareness, making it a strong candidate for inclusion in the classification task, where the distinction between unidirectional and bidirectional completions may be a key feature for the classifier.

RoBERTa, a variant of BERT, improved upon BERT's pretraining strategies by using larger batches and more extensive datasets. Liu et al.'s work in [5] highlights the importance of optimization in LLM performance. RoBERTa's improvements in efficiency and performance, particularly in sentence encoding, make it a strong model for generating coherent text completions. As a classifier, distinguishing between BERT and RoBERTa may require attention to subtle differences in their completion styles, likely influenced by the additional pretraining steps.

Li et al.'s work in [6] provides a comprehensive overview of the methods and challenges involved in neural text generation. This paper is highly practical, offering insights into the architectures behind text generation models, which can be directly applied to the sentence completion aspect of the current task. It also discusses common challenges such as repetition, coherence, and diversity, which are crucial for analyzing how different LLMs generate text and how a classifier could differentiate between outputs based on these traits.

Clark et al.'s paper [2] explores how BERT attends to different parts of a sentence during the encoding process. This research is relevant for understanding the mechanics behind sentence completions generated by BERT-based models. By analyzing how attention mechanisms influence sentence structure, this paper offers valuable insights for identifying patterns in sentence completions. These patterns, such as how much weight BERT gives to certain words or phrases, may help differentiate BERT-based completions from other models in the classification task.

Fine-tuning LLMs for specific tasks is a critical step in applying pretrained models to real-world applications. Mosbach et al.'s work in [4] focuses on the technical aspects of fine-tuning, which directly impacts how models like GPT, BERT, and T5 behave when completing sentences. Understanding the fine-tuning process is essential, as models fine-tuned on specific datasets may exhibit unique traits in their text completions. The paper's exploration of weight initialization and data ordering can inform how fine-tuning affects model behavior, helping to refine the classifier's ability to detect differences between LLMs.

3 Dataset Curation

For this project, tweets of Elon Musk were scraped using Twitter API v2, resulting in a dataset of 12,042 tweets. The reason for selecting this dataset was to check how different LLMs perform text completion on contemporary information even without context. The raw data underwent extensive cleaning to eliminate unwanted elements, such as "Replying to," standalone numbers, URLs, mentions, emojis, and retweet markers. Additionally, special characters were removed, and tweets containing fewer than five words were filtered out. After the cleaning process, the tweets were truncated to their first three words (xi), which served as prompts for various large language models (LLMs) to generate text completions. The completed sentences (xj) are stored in a new csv. The LLMs we used in this project are GPT-2, GPT-Neo, Facebook OPT, Gemma2:2B and Mistral-OpenOrca.

3.1 Hugging Face Models

We obtained GPT-2, GPT-Neo and Facebook OPT models from Hugging Face. Owing to their relatively small size we downloaded the offline versions of these models provided by Hugging Face. GPT-2 even though its not up to date with world data, has very good contextual understanding and hence making it a good choice. GPT-Neo, being a lighter model compared to GPT-2, one would expect bad performance from GPT-Neo but the results are almost comparable to GPT-2.

Facebook OPT, Trained on a diverse range of internet text, its architecture enables it to produce high-quality, contextually appropriate text completions. It is one the best available model in its parameter size.

3.2 Ollama Models

For other models with parameters in the order of billions, we do not have the required computational resources. Hence we used Ollama. Ollama makes powerful language models easily accessible and user-friendly. We obtained the text completion from Gemma2:2B and Mistral-OpenOrca

Both these models are powerful LLMs which are capable of taking prompts and responding to them.

3.3 Models Considered but Not Used

We also considered additional models from Ollama, specifically LLaMa, Neural Chat, and Phi3. However, we did not include these model due to the following reasons:

LLaMa: While capable, LLaMa exhibited inconsistent results, occasionally repeating input as output or failing to produce any completion. This unpredictability rendered it unsuitable for our requirements. Neural Chat and Phi3: Although these models generated good outputs, they required excessively high computational resources, taking over 12 hours to run. Such impractical processing times led us to prioritize models that offered a better balance between performance and efficiency.

Out of the 12,042 sentences were scraped for this project, only 4,000 sentences were used for text completion due to the computational limitations mentioned above. Each of the LLM output 4000 text completion, making the entire dataset of length 20000 sentences. This careful selection of models and data ensures that the project remains feasible while achieving the desired outcomes in text completion.

4 Classifier and Training

The entire training pipeline is showed in figure 2. After the preprocessing is done, the training process involves 2 steps. The first step is embedding for which SBERT is used. The input sentences are tokenized, and embeddings are generated from the model outputs through mean pooling, which condenses the embeddings to a manageable size. You can see the pipeline of S-Bert in 1. The classifier processes these embeddings, with dropout layers implemented to enhance generalization. Each batch of data is embedded separately using this architecture, resulting in embeddings with 768 dimensions. In the second step, a Simple Fully Connected (FC) classifier is employed, utilizing the Adam Optimizer. The model architecture consists of two fully connected layers, along with a dropout layer (with a dropout probability of 0.3) to mitigate overfitting. The loss is calculated using cross-entropy loss.

For preprocessing, initially, the CSV files corresponding to five different large language models (LLMs) are loaded into a combined DataFrame, with each entry tagged by its respective model. For our best model, The training process is configured to run for 15 epochs with checkpoint intervals set at 5 epochs, and a learning rate of 0.001 is established. The labels for each model are then encoded to facilitate training, and the dataset is split into training and testing subsets using an 80:20 ratio. The training and testing datasets are created and loaded in batches, with a specified batch size of 32.

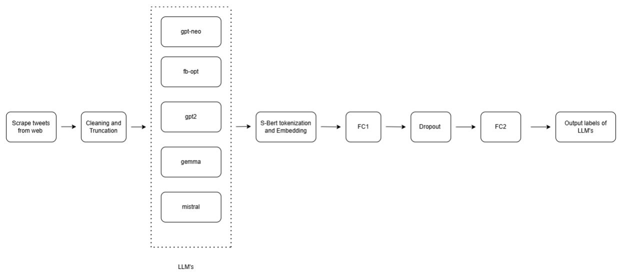

Figure 2: Model Architecture

Throughout the training loop, the model's performance is assessed by calculating the loss and adjusting the weights accordingly through backpropagation. After each epoch, the model is evaluated on both training and testing datasets to monitor performance metrics and ensure effective learning. Finally, checkpoints are created during the training process, allowing for model states to be saved and loaded as needed. At the conclusion of the training, a final evaluation is performed, producing a classification report to assess the model's accuracy and overall performance.

5 Ablation Studies

The experimental results highlight several trends in the performance of a classifier as various parameters were adjusted. The model with 15 epochs, a learning rate of 10^{-3}, drop out of 0.3 and a batch size of 32 achieved a final test accuracy of 70.75%. As the number of epochs was increased to 25 while keeping other parameters constant, the test accuracy dropped to 69.95%. This decline suggests potential overfitting, as the training accuracy continued to rise significantly, indicating the model was fitting the training data well but failing to generalize effectively to the test set. Conversely, reducing the learning rate to 10^{-4} resulted in a marked decrease in performance, yielding a final test accuracy of only 62.75%. The lower learning rate hampered the model's ability to converge, leading to suboptimal learning during training. On the other hand, increasing the learning rate to 10^{-2} produced a modest improvement with a test accuracy of 66.05%, but still did not reach the performance of the base model, implying instability in training at higher learning rates. Notably, increasing the dropout rate to 0.5 led to an improvement (compared to the previous result) in test accuracy to 70.47%, indicating that a higher dropout rate effectively reduced overfitting and improved generalization. The introduction of an additional fully connected layer and normalization slightly increased the test accuracy to 70.53%, suggesting that architectural enhancements can capture more complex patterns in the data. However, decreasing the batch size to 16 resulted in a test accuracy of 69.97%, indicating that while smaller batches might offer some benefits in terms of regularization, they may also hinder learning dynamics. Also this may be attributed to the embeddings of S-Bert because S-Bert embedds a batch of sentences and reducing the batch size reduces the effectiveness of the embedding. Overall, these experiments highlight the importance of carefully tuning hyperparameters such as epochs, learning rates, dropout rates, and model architecture to optimize classifier performance in machine learning tasks.

Table 1: Ablation Study Results

Params Changed	**Accuracy**
Best Model	**70.75%**
Increased epochs to 25	69.95%
Learning rate decreased to 10^{-4}	62.75%
Learning rate increased to 10^{-2}	66.05%
Dropout rate increased to 0.5	70.47%
Added FC layer and normalization layer	70.53%
Decreased batch size to 16	69.97%

6 Results

The final test accuracy achieved by the model was **70.75%**, indicating a strong performance in classifying the data. A heatmap was generated to visualize precision, recall, and F1-scores for each LLM 5. The classification metrics for each model are as follows: Precision: The model performed best for Gemma2:2B, achieving a precision of **0.77**. In contrast, GPT-Neo exhibited the lowest precision at **0.65**.

Recall: Gemma2:2B again led with a recall of **0.82**, indicating that the model effectively captured most instances of this category. However, GPT-Neo had a recall of **0.64**, suggesting that some true instances were missed in this model.

F1-score: Gemma2:2B achieved the highest F1-score of **0.79**, while GPT-Neo had the lowest at **0.65**. The overall performance metrics indicate that the model is particularly proficient at identifying instances of GPT-2 and Gemma2:2B, while performance for GPT-Neo requires improvement.

These results underscore the effectiveness of the model in capturing the nuances of the data but also highlight areas for future enhancement, particularly for models that exhibit lower precision and recall.

Figures 4 and 3 show the accuracies(both train and test) and loss vs epochs respectively.

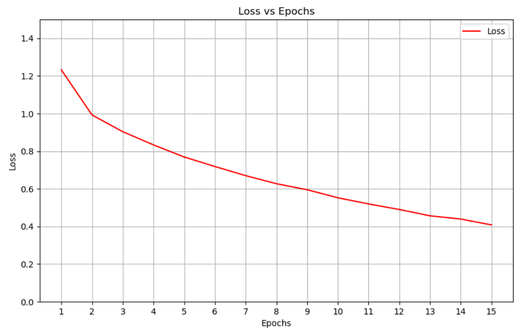

Figure 3: Epochs vs Loss

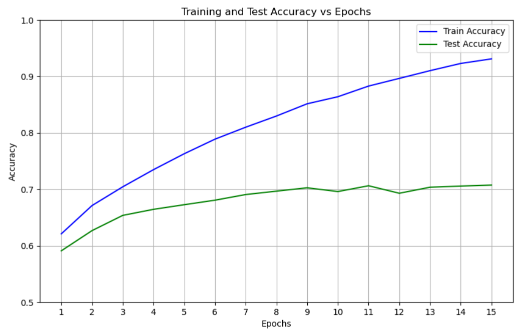

Figure 4: Train and Test Accuracy vs Epochs

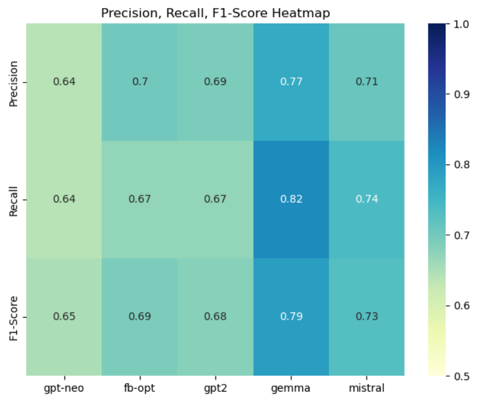

Figure 5: Precision, Recall and F1 for 5 LLM's

7 Conclusion and Future Work

In this work, we are able to distinguish 5 LLMs with an accuracy of 70.75%. In future work, we can perform more in depth analysis of the style of text generation of different LLMs.

References

[1] Tom B. Brown et al. "Language Models are Few-Shot Learners". In: *CoRR* abs/2005.14165 (2020). arXiv: `2005.14165`. URL: `https://arxiv.org/abs/2005.14165`.

[2] Kevin Clark et al. "What Does BERT Look at? An Analysis of BERT's Attention". In: *BlackboxNLP@ACL*. 2019. URL: `https://api.semanticscholar.org/CorpusID:184486746`.

[3] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR* abs/1810.04805 (2018). arXiv: `1810.04805`. URL: `http://arxiv.org/abs/1810.04805`.

[4] Jesse Dodge et al. "Fine-Tuning Pretrained Language Models: Weight Initializations, Data Orders, and Early Stopping". In: *CoRR* abs/2002.06305 (2020). arXiv: `2002.06305`. URL: `https://arxiv.org/abs/2002.06305`.

[5] Yinhan Liu et al. "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: *CoRR* abs/1907.11692 (2019). arXiv: `1907.11692`. URL: `http://arxiv.org/abs/1907.11692`.

[6] Ziang Xie. "Neural Text Generation: A Practical Guide". In: *CoRR* abs/1711.09534 (2017). arXiv: `1711.09534`. URL: `http://arxiv.org/abs/1711.09534`.