Midterm Project: CSE 584

Archisman Ghosh
CSE Department
Penn State University
State College, PA, USA
apg6127@psu.edu

***Abstract*—This project investigates the task of identifying which Large Language Model (LLM) generated a given text completion. Given a set of truncated text fragments x_i and their corresponding completions x_j, produced by three different LLMs—GPT-2, GPT-Neo, and BLOOM, we build a deep learning-based classifier to predict the LLM responsible for a given text pair (x_i, x_j). The same input x_i is provided to each of the three models, resulting in distinct completions x_j, which exhibit subtle variations in style, vocabulary, and structure. To classify the LLMs that generated the completions, we employ an LSTM-based classifier that leverages both the input and completed text to capture the nuances of the output of each model. Our classifier achieves a test accuracy of $\sim 60\%$, demonstrating the feasibility of distinguishing between different LLMs using deep learning techniques. We also propose an extension of the idea by experimenting with pre-trained LLMs to classify the generated text.**

***Index Terms*—Large Language Models, Long-Short Term Memory, Multi-class Classification**

I. Introduction

Large Language Models (LLMs) like GPT-3/4, BLOOM, and LLaMA among many, have shown tremendous capabilities in generating coherent and contextually relevant text [1]. These models are trained on vast and diverse datasets and have different architectures, resulting in distinct characteristics in the text they produce. These differences, althjough subtle, provide an opportunity to study and potentially identify which model generated a particular text completion. In this project, we focus on developing a classifier to distinguish between text completions generated by different LLMs. Using a dataset composed of pairs of truncated text x_i and their corresponding completions x_j generated by three models (GPT-2, GPT-Neo, and BLOOM), we employ a deep learning-based approach to predict the LLM responsible for each completion.

A. Motivation

As LLMs become increasingly integrated into applications across various industries, the ability to identify which model generated a given text becomes critical. This is especially important for responsible use of AI, which can be enforced by model auditing, and even detecting AI-generated content in sensitive domains such as news, education, and social media. With the rapid development of new LLMs, understanding their differences in text generation can help improve transparency and accountability in AI systems. Furthermore, classifying LLMs based on generated text can provide insights into the limitations and biases of individual models, helping users select the most appropriate models for specific tasks.

B. Contributions

The major contributions of this study are as follows:

1) We compose a dataset of truncated texts and their corresponding completions generated by three different LLMs–GPT-2, GPT-Neo, and BLOOM.
2) We develop an LSTM-based classifier to distinguish between the LLMs from the text generated by them.

C. Report Structure

Section II provides a background and related works. Section III covers the dataset. Section IV describes the procedure and architecture of the LSTM-based classifier. Section V presents the results. We discuss the results in Section VI and conclude in Section VII.

II. Background

A. Large Language Models

Large Language Models (LLMs) are deep learning models trained on vast amounts of text data, capable of performing a wide range of natural language processing (NLP) tasks including text generation, summarization, and translation. LLMs capture complex linguistic patterns and dependencies by processing data in parallel across multiple layers of self-attention mechanisms of transformers. Text generation, in particular, is one of the most remarkable capabilities of LLMs, enabling them to generate coherent, contextually relevant, and sometimes creative completions based on a given input or prompt. Text generation in LLMs is based on the ability of models to predict the next word (or token) in a sequence, conditioned on the preceding context. By training on enormous datasets with diverse linguistic patterns, these models learn the statistical properties of language, which allows them to generate fluent and contextually appropriate text when given a prompt. LLMs such as GPT-2/3 [2], and BLOOM (BigScience Large Open-science Open-access Multilingual Language Model) [3] have demonstrated state-of-the-art performance in text generation, producing outputs that often resemble human-written text. This ability to generate text with a high degree of coherence and relevance has led to the widespread adoption of LLMs across industries for tasks like conversational agents, content creation, and code generation. GPT models, developed by

OpenAI, are autoregressive LLMs known for their powerful text generation capabilities. GPT-2, with 1.5B parameters, and GPT-3, with 175B parameters, generate coherent and contextually relevant text across domains. BLOOM, developed by BigScience, is an open-access multilingual model with 176 billion parameters, supporting 46 languages. BLOOM offers similar text generation capabilities to GPT but focuses on multilingualism and open science.

B. Long Short-Term Memory Networks

LSTM Networks (Long Short-Term Memory Networks) [4] are a type of recurrent neural network (RNN) that are particularly effective at learning from sequential data, such as text, time series, or speech. LSTMs solve a major problem of vanishing gradient which makes it capable to learn long-term dependencies in sequences, by using specialized gates (input, forget, and output gates) that regulate the flow of information, allowing the network to retain important information over long distances while forgetting irrelevant details. LSTMs are particularly useful in the context of classification tasks because they can capture the temporal or contextual relationships between elements in a sequence. For example, when dealing with text data, the meaning of a word often depends on its context, i.e., the words that come before and after it. LSTMs are capable of remembering this context over long sequences, making them highly effective for tasks like sentiment analysis, language modeling, and text classification. In text classification, LSTMs can process the input text word by word, maintaining a memory of important features that inform the classification decision. This ability to learn both short-term and long-term dependencies makes LSTMs particularly useful in classifying complex and sequential data, such as distinguishing between the stylistic or semantic differences in text completions generated by different models.

C. Related Work

Several recent works have explored different aspects of large language models (LLMs) and their applications in text classification and detection tasks. LLM-DetectAIve presents a fine-grained machine-generated text detection system, focusing on distinguishing between various levels of LLM involvement, such as fully machine-generated versus machine-polished texts. This research highlights the challenges of detecting human-like outputs from advanced LLMs like GPT-4 and Claude, particularly for educational settings where such interventions may be prohibited [5]. There have been works that explore the cost-effectiveness of using LLMs for few-shot classification in resource-limited domains like finance. By leveraging in-context learning with models like GPT-3.5 and GPT-4, it demonstrates how LLMs can outperform traditional fine-tuned models in few-shot scenarios while also offering methods for cost reduction through techniques like retrieval-augmented generation [6]. Another study investigates the potential of using LLM-generated labels for supervised classification, comparing the performance of models like BERT and RoBERTa fine-tuned with human versus LLM-generated labels. It suggests that fine-tuning supervised models with LLM-generated labels offers a fast and cost-effective alternative to human annotation, though challenges like noise and label consistency must be addressed [7].

III. Dataset

A. Truncated Text

To create the truncated texts x_i, we utilize the WikiText-2 dataset as the source for generating incomplete text fragments. The WikiText-2 dataset is a popular text corpus for language modeling tasks, providing clean and well-structured English sentences extracted from Wikipedia articles. The dataset is publicly available through the Hugging Face `datasets` library. For the prototype dataset for the project, we truncate the sentence to retain the first 5 words only. We consider 1000 such unique samples to prepare a large enough collection of x_i, which would help in the classification of the LLMs based on the completion of these truncated texts.

B. Compiled dataset

After generating truncated texts x_i from the WikiText-2 dataset, we generate corresponding x_j from different LLMs to compile a complete dataset to classify between the LLMs. Every entry of the dataset is a triple $(x_i, x_j, model)$. The x_i values are passed to the LLMs as a prompt and the model generates a continuation of the input text using the `generator` object. We also specify the LLMs to generate sentences of length 50. All models are open-sourced and available on Hugging Face.

IV. Procedure

A. Large Language Models

We consider three state-of-the-art LLMs for the comparison in our study.

GPT-2: This is a widely-used LLM with 1.5B parameters. It is known for generating coherent and contextually relevant text completions. Trained on a diverse corpus, GPT-2 is versatile across various natural language processing tasks like text generation, summarization, and translation.

GPT-Neo: GPT-Neo is created by EleutherAI and has 3 main versions with 125M, 1.3B, and 2.7B parameters. For our experiments, we us ethe 2.7B parametered GPT-Neo. It is an open-source alternative to GPT-3, designed to handle large-scale NLP tasks.

BLOOM: BLOOM is a multilingual language model developed by the BigScience project, with sizes ranging from 560M to 176B parameters. We use the 1.7B version for this project. Trained on a massive multilingual dataset, BLOOM excels in generating text across diverse languages and contexts, making it suitable for varied language generation tasks.

B. Classifier Architecture

We design a Sequential neural network model for text classification. The model consists of an embedding layer followed by two LSTM (Long Short-Term Memory) layers (Table I). The embedding layer maps each word in the vocabulary to a dense vector representation of size 64. The two LSTM layers have 128 units each; the first LSTM layer returns the entire sequence of hidden states, while the second one only returns the final output. After the LSTM layers, a Dropout layer with a 50% dropout rate is used to prevent overfitting, followed by a Dense layer with 64 neurons and a ReLU activation function. The output layer consists of a Dense layer with a softmax activation to handle the multi-class classification problem, with 3 units (corresponding to the three LLMs used to generate the dataset).

TABLE I
LSTM-BASED CLASSIFIER

Layer Type	Output Shape	Parameters
Embedding	(_, 50, 64)	640,000
LSTM (1st)	(_, 50, 128)	99,840
LSTM (2nd)	(_, 128)	131,584
Dropout	(_, 128)	0
Dense (ReLU)	(_, 64)	8,256
Dropout	(_, 64)	0
Dense (Softmax)	(_, 3)	195

C. Tokenization

Before training the classifier, the text data is tokenized using the Tokenizer from the Keras preprocessing module of TensorFlow. The tokenizer is set with a maximum vocabulary size of 10,000 words, and an "Out of Vocabulary" (OOV) token is used for words not found in the vocabulary. Texts are first converted into sequences of integers, where each word in the text is mapped to a unique integer representing its position in the word index of the tokenizer. To ensure uniform input to the LSTM layers, all sequences are padded or truncated to a fixed length of 50 tokens, making each input sequence consistent in length. Padding is applied at the end of the sequences to maintain the original order of words.

D. Training

The model is trained using the Adam optimizer with a learning rate of 10^{-5}, which is tuned to ensure stability during training. The loss function used is sparse categorical cross-entropy, which is appropriate for multi-class classification tasks where the target labels are integers. The training is conducted for 50 epochs with a batch size of 32, and the data is split into training and validation sets with an 80/20 ratio. To further prevent overfitting, a 50% dropout is applied after the LSTM and dense layers. We evaluate the efficiency of the model by the test and validation accuracy and loss (Fig. 1)

V. RESULTS

A. Simulation Setup

The LLMs were inferenced on Google Colab using an A100 GPU to obtain faster results for the completed dataset. The classifier was implemented using TensorFlow libraries and run on a machine with 16GB RAM on an Intel Core i7-6700 CPU at a clock frequency of 3.40 GHz.

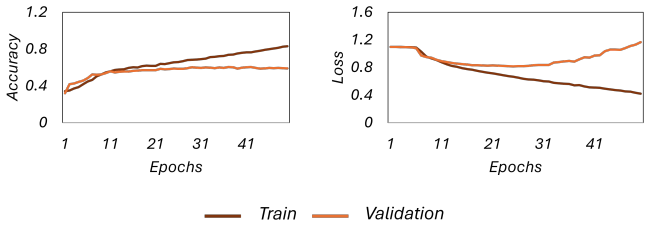

Fig. 1. Plot representing the trend in classification accuracy and loss for the train and validation set. We can observe that the peak train accuracy is ∼82% and there is a significant decline in the loss plots, describing the efficacy of the proposed classifier.

B. Classification

We present the trends in classification in Fig. 1. From the figure we can observe that there is good convergence in the training loss proving that the classifier works well for the dataset in classifying the LLMs based on the completion texts. The loss function used in the classifier is sparse categorical cross-entropy. It computes the negative log-likelihood of the true class, meaning it penalizes the model based on how far the predicted probability for the correct class is from 1. Sparse categorical cross-entropy is calculated as

$$Loss = -log(p_{true})$$

where p_{true} is the predicted probability assigned by the model to the correct class.

VI. DISCUSSION

In this section, we discuss the approach to the problem, potential improvements, and future research.

A. Dataset

While WikiText-2 is a strong dataset for generating truncated text x_i due to its clean and well-structured language from Wikipedia, other datasets could offer better variety and complexity. OpenWebText (an open-source version of the WebText corpus used to train GPT-2) would provide more diverse and less formal language, simulating a broader range of internet text. Common Crawl could introduce a massive, diverse corpus with web content from various domains. For domain-specific language, BookCorpus (narrative text) or Reddit conversations (colloquial, conversational text) could offer more stylistic variation, helping models generalize to more complex or informal text generation tasks. Also, the sample size could be increased to obtain a more diverse dataset for the LLMs to compile.

B. LLMs

The current idea presents a comparison between GPT-2, GPT-Neo, and BLOOM. The choice of LLMs in the comparison is limited by the price of accessing them as well as the resources required in inferencing them for the generation of the completed dataset. For example, GPT-3/4 ($\sim$ 175B parameters), LLaMA (7-65B parameters), PaLM (8-540B parameters) among other LLMs can be used to compile a richer dataset and perform better classification.

C. Classifier

An LSTM-based classifier was chosen due to its ability to capture long-term dependencies in sequential data, which is crucial for analyzing text completions. Unlike feed-forward networks, LSTMs retain information across sequences, allowing them to model the context and relationships between words over time. This is essential for distinguishing subtle differences in style or structure between text completions generated by different LLMs. LSTMs are highly effective for text classification tasks, making them suitable for this project, where the focus is on recognizing patterns across entire sequences rather than isolated tokens.

We also perform an additional set of experiments using the pre-trained BERT (Bidirectional Encoder Representations from Transformers) model [8]. Using a pre-trained BERT model would be better because BERT captures bidirectional context, understanding both preceding and following words, which enhances its ability to recognize nuanced patterns in text completions. Fine-tuning BERT leverages its pre-trained knowledge, leading to improved classification accuracy and efficiency, especially for distinguishing subtle differences between the LLMs. However, we could perform only partial experiments due to the lack of resources.

1) BERT-base: This model has $\sim$110M parameters with 12 transformer layers and 12 attention blocks.

2) BERT-large: This model has $\sim$340M parameters with 24 transformer layers and 16 attention blocks.

3) Training details: We perform the experiments with similar hyperparameters as the LSTM-based classifier, using an Adam optimizer with a learning rate of 2e-05, for 2 epochs with a batch size of 16 and a loss function of sparse categorical cross-entropy. We also use the `BertTokenizer` to convert the text into tokenized sequences, where each word or subword is mapped to a unique token ID of length 512 (better for longer sequences).

4) Results: We show in Fig. 2 that the values of test accuracy for BERT-based classifier models are as good as the LSTM-based classifier, the only caveat being that the BERT-based models were trained for 2 epochs. In our defense, BERT being pre-trained on large data, needs to be fine-tuned for only a few epochs to implement it for a downstream task. Therefore, on further experimentation and hyperparameter tuning, BERT and other similar pre-trained models should be able to perform better than LSTM-based classifiers given a larger and richer dataset and sufficient resources.

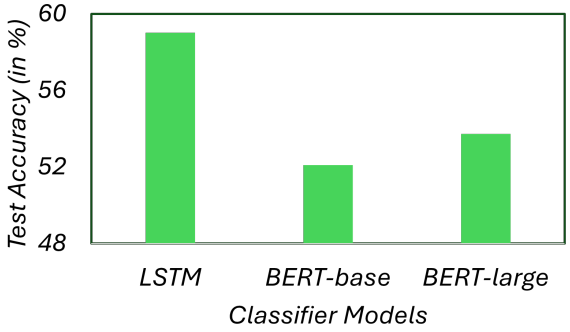

Fig. 2. Plot comparing the test accuracy between LSTM-based classifier and the BERT models. We see that the LSTM-classifier achieves a test accuracy of $\sim$ 58% and both the BERT models achieve over 52% accuracy but with very less fine tuning. On increasing the amount of fine tuning the hyperparameters, we could observe a potential improvement in the BERT-based classifiers.

D. Future Work

To improve on this project idea, we could extend the experimentation with other transformer models like RoBERTa [9], T5 [10], or DistilBERT [11]. These models have different training objectives or architectures that may provide better performance or efficiency for text classification tasks. Even GPT-2/3/4 or PaLM [12] could provide insights into the impact of using autoregressive models for classification tasks. The dataset could be extended to incorporate larger amounts of data from diverse text sources to generalize the classifier better.

VII. Conclusion

This project explores the classification of text generated through models like GPT-2, GPT-Neo, and BLOOM and demonstrates the potential for distinguishing between different LLMs using simple deep learning models like LSTM-based classifiers using the stylistic or structural characteristics of their text completions. Various works in the field, including machine-generated text detection and cost-efficient LLM utilization, emphasize the growing importance of these models across diverse NLP tasks. Future work could expand on this by incorporating additional transformer models and extending datasets for better generalization. The use of pre-trained models like BERT further highlights the efficiency of fine-tuning for classification tasks, reinforcing the value of LLMs in modern AI applications.

References

[1] Rongwu Xu, Zehan Qi, Zhijiang Guo, Cunxiang Wang, Hongru Wang, Yue Zhang, and Wei Xu. Knowledge conflicts for llms: A survey, 2024.

[2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[3] BigScience Workshop et al. Bloom: A 176b-parameter open-access multilingual language model, 2023.

[4] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[5] Mervat Abassy, Kareem Elozeiri, Alexander Aziz, Minh Ngoc Ta, Raj Vardhan Tomar, Bimarsha Adhikari, Saad El Dine Ahmed, Yuxia Wang, Osama Mohammed Afzal, Zhuohan Xie, Jonibek Mansurov, Ekaterina Artemova, Vladislav Mikhailov, Rui Xing, Jiahui Geng, Hasan Iqbal, Zain Muhammad Mujahid, Tarek Mahmoud, Akim Tsvigun, Alham Fikri Aji, Artem Shelmanov, Nizar Habash, Iryna Gurevych, and Preslav Nakov. Llm-detectaive: a tool for fine-grained machine-generated text detection, 2024.

[6] Lefteris Loukas, Ilias Stogiannidis, Odysseas Diamantopoulos, Prodromos Malakasiotis, and Stavros Vassos. Making llms worth every penny: Resource-limited text classification in banking. In *Proceedings of the Fourth ACM International Conference on AI in Finance*, ICAIF '23, page 392–400, New York, NY, USA, 2023. Association for Computing Machinery.

[7] Nicholas Pangakis and Samuel Wolken. Knowledge distillation in automated annotation: Supervised text classification with llm-generated training labels, 2024.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[9] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[10] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023.

[11] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020.

[12] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways, 2022.