CSE 584 Midterm Project

Siddharth Anmalsetty & Avitej Iyer

Abstract

This project aimed to determine which of a set of LLMs a text sample was generated from. For example, some prewritten prompt xi can be inputted into an LLM and some output xj will be returned. This data was then collated as 20 responses to 10 prompts from 5 different LLMs for 1000 data points. This was passed through three different classifier methods: a logistic regression model, a custom LSTM model, and a pre-trained BERT model. The accuracies of each method are as follows: 37%, 54.5%, and lastly 75.5%. These results were not excellent but, given the limitations of the dataset, are within expectations.

Dataset Curation

Model Choices

The first thing decided on was the LLMs used for testing. Given its notoriety as the frontrunner of LLM technology, ChatGPT-4o was an obvious choice. In the interest of creating a more varied data set, we wanted to use LLMs that were more analytical in their responses as well as ones that were a bit more conversational. We tested a few models and found that models like Claude (a model that is designed to be very ethical/forthright) and Gemini returned quite short and simple responses, while GPT and Cohere were quite descriptive and creative in their responses. In the interest of adding another very relevant model, we also added Microsoft Copilot. With this in mind, we moved on to the next step.

Prompt Generation

The second thing that was done was to generate the prompts. The first prompt we decided to use was the one given in the assignment description "Yesterday I went". Given the tone of this, we decided to generate more conversational prompts. The ones created were as follows:

"Yesterday I went"
"Tomorrow I will"
"I spoke to my friend about"
"I went to the store and bought"
"My favorite activity is"

We also wanted to test prompts that would have a bit more variance and open-ended to see whether there were any significant differences in how the LLMs responded to these questions. The prompts decided upon were as follows:

"The two things I find the most different are"

"The thing I find most valuable is"

"One thing I would change about the world is"

"I would define creativity as"

"A fulfilling life includes"

We believed 5 conversational prompts and 5 open-ended prompts were enough to generate a strong sample to see what differences would arise in how the model was trained.

Data Collection

In the interest of getting the data in the most unbiased way, each different model was prompted with the same initial and subsequent queries. The initial query used was:

"complete the prompt "XXXXXX" in 20 different ways"

Where "XXXXXX" was one of the ten prompts we decided. Subsequent queries were sent as just the prompt:

"XXXXXX"

The data was collected in three columns: Prompt, Response, and Model. 1000 data points were collected in total and 80% (800) was used as the training set. Given this, all data was collected by manual prompting and input into a spreadsheet manually. Responses were collected by entering the identical prompts into each LLM's interface and copying the responses directly into a spreadsheet. To minimize bias as well, prompts were given in a randomized order to avoid bias from sequential queries. This data can be found in the GitHub. This dataset is limited, but without access to premium accounts for any of these LLMs, we were unable to use model APIs to generate a much larger amount of data.

Classifier

Three different classifiers were used and will be discussed below. Firstly, it was decided to concatenate the prompt and response together to simplify implementation and reduce complexity. This was done via pandas after uploading our data as a CSV.

Logistic Regression

This first model was made as a proof of concept. This was to serve as a baseline for testing results to gauge the performance as compared to a simple implementation. The first step is to convert the text data into numerical data using TF-IDF (Term Frequency-Inverse Document Frequency) vectorization. This allowed for each word to be considered separately as associated with their input and then trained on this data to see whether or not the model could be determined from that data. We used a maximum of 5000 input features as that would cover the majority of the unique/important words in our dataset of 1000 terms. The intent was to find out what words correspond most strongly to which model is being used. Since Logistic Regression works where the optimization is done in one pass, the model is trained, and then the optimization finishes. As there are no epochs to track, that is why it cannot be found in the graph in Figure 6.

LSTM

LSTM(Long Short-Term Memory) models are types of recurrent neural networks that are used in language processing that work by keeping track of relevant information using gates and forgetting irrelevant data. This can allow our model to keep track of important info for quantifying the differences between our output sequences.

The data is first tokenized and then preprocessed to be padded to equivalent lengths of 35 words at most. This was the longest sequence in our dataset. These were then passed through an embed layer to transform the tokens into dense vectors. After this, they were passed through an LSTM layer to capture important textual information. Next was a dense layer using a ReLU activation function. The ReLU activation helps introduce non-linearity, which enables the model to learn complex patterns. The number of units (128) is again chosen to keep the model moderately complex without overfitting. This is then followed by another Dense layer to move into our 5 output classes using a SoftMax activation function. The softmax activation function is chosen because it's ideal for multi-class classification tasks, as it converts the raw output logits into a probability distribution over the classes. Dropout layers were inserted between the output layers to hopefully improve overfitting. A dropout of 0.3 was selected with just trial and error - it seemed to provide the best results. We went with an initial dropout of 0.5 (because of a small dataset), but this proved to be too aggressive. The following is a list of the layers in order.

Layers
Embed
LSTM
Dropout
Dense
Dropout_1
Dense_1

Finally, we used "Adam" as our optimizer because it combines the benefits of both momentum and RMSprop optimizers. It works well in most cases without much tuning. Adam adjusts the learning rate dynamically, making it a good choice for smaller datasets like ours. Data was trained for 150 epochs as this came to return the highest accuracy from experiments.

BERT

The last classifier we wanted to use was a pre-trained classifier to compare our results to a state-of-the-art model. BERT uses a transformer to strongly learn the relationships between text and their classes. We first passed our tokenized input into the model. We used the "base" version of BERT - it has 12 transformer layers, 12 attention heads, and a hidden size of 768, with around 110 million parameters. This version provides a good balance between computational efficiency and performance for most NLP tasks, especially on smaller datasets. Using the larger "bert-large-uncased" would have been overkill for us. We chose a max sequence length of 128 because a length of 128 is often sufficient to capture meaningful content in most sentence-level text tasks. A smaller batch size (like 16, in our case) was chosen to balance memory constraints while still allowing for stable gradient updates. Larger batch sizes can provide more stable training, but on a limited dataset, batch sizes of 16 or 32 are typical for fine-tuning. Since BERT is well-trained, a large learning rate could make it "unlearn" the pre-trained knowledge. Therefore, we used a learning rate of 5e-5. Finally, we used the AdamW optimizer because the weight decay (L2 regularization) prevents the model from overfitting by penalizing large weights. BERT was only trained for 15 epochs for the same reason. Its performance was far above the LSTM classifier which was expected.

Results

Figure 1: Test Accuracy for All Models

Logistic Regression	*LSTM*	*BERT*
37.00%	54.5%	75.5%

Figure 2: Comparison of Test Accuracies

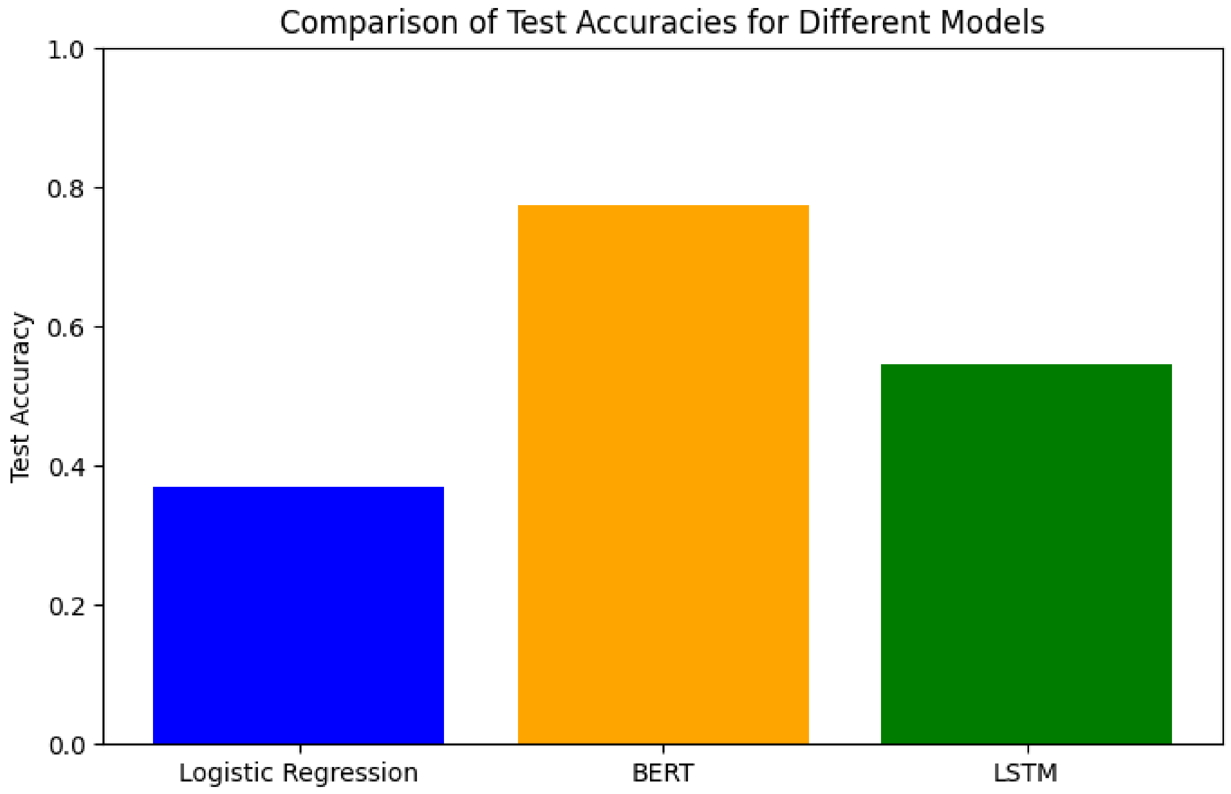


Table 1: Accuracy Report for Logistic Regression

Model	Precision	Recall	F1-Score	Support
ChatGPT-4o	0.32	0.36	0.34	33
Claude	0.25	0.15	0.18	48
Cohere	0.66	0.61	0.63	38
Copilot	0.22	0.14	0.17	44
Gemini	0.36	0.7	0.47	37
Accuracy			**0.37**	200
Macro Avg	0.36	0.39	0.36	200
Weighted Avg	0.35	0.37	0.35	200

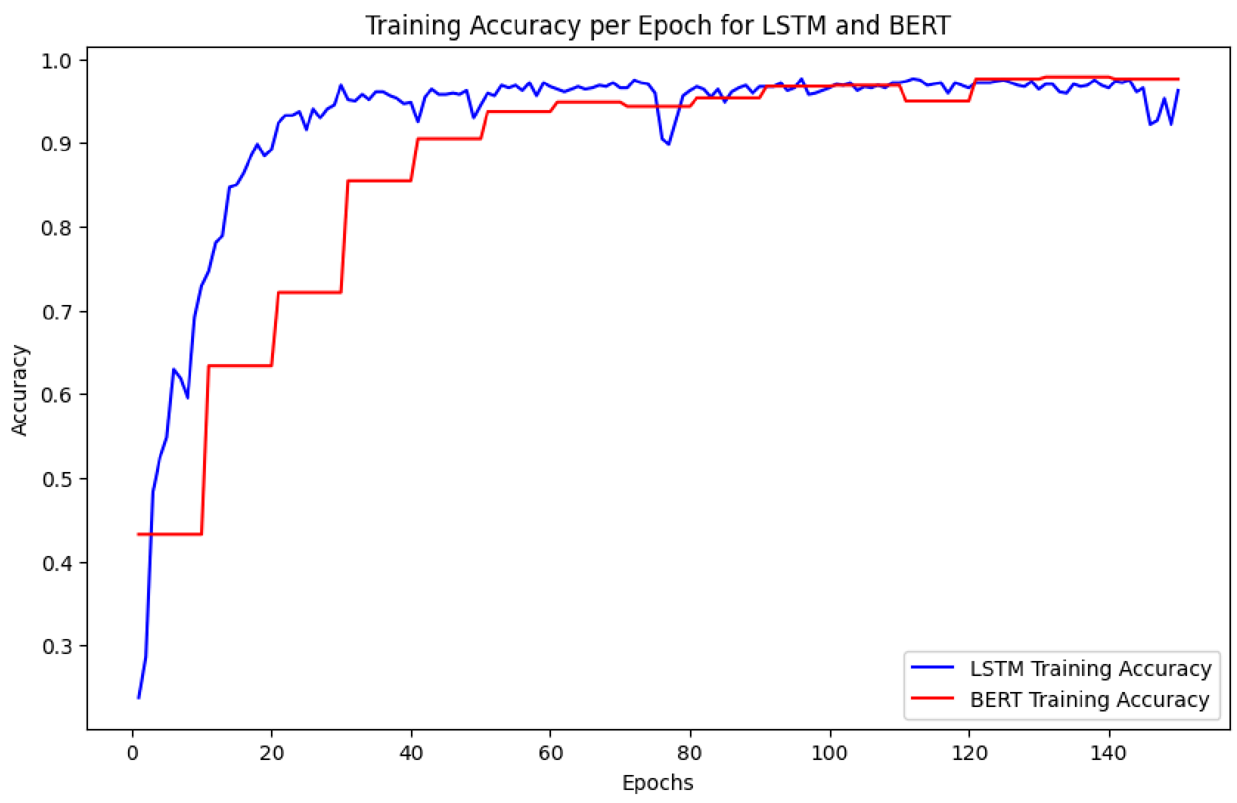
Table 2: Accuracy Report for Bert

Model	Precision	Recall	F1-Score	Support
ChatGPT-4o	0.72	0.79	0.75	33
Claude	0.80	0.58	0.67	48
Cohere	0.86	0.95	0.90	38
Copilot	0.94	0.75	0.84	44
Gemini	0.62	0.86	0.72	37
Accuracy			**0.78**	200
Macro Avg	0.79	0.79	0.78	200
Weighted Avg	0.80	0.78	0.77	200

Table 3: Accuracy Report for LSTM

Model	Precision	Recall	F1-Score	Support
ChatGPT-4o	0.57	0.64	0.6	33
Claude	0.45	0.52	0.48	48
Cohere	0.81	0.68	0.74	38
Copilot	0.47	0.34	0.39	44
Gemini	0.51	0.59	0.55	37
Accuracy			**0.55**	200
Macro Avg	0.56	0.56	0.55	200
Weighted Avg	0.55	0.55	0.54	200

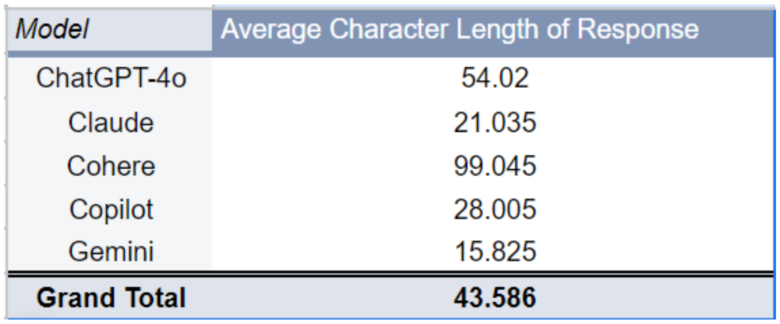
Figure 3: Training Accuracy per Epoch for LSTM and BERT



Analysis

Dataset Observations

Something to consider when looking at the performance of the model is how strong the initial data is. When taking a human pass over the data some trends are easily found. For example, Cohere had extremely long and detailed responses compared to the others. For the prompt "I spoke to my friend about", Cohere responded with "a difficult decision they were considering, offering my perspective and helping them carefully weigh the pros and cons to find clarity," while Gemini responded with "their favorite music". This brings up a concern because the responses are widely different sizes which may cause an overfit to length rather than other characteristics. This could also be a valid way to differentiate between these models. Below is each model and the average length of their responses.



Model	Average Character Length of Response
ChatGPT-4o	54.02
Claude	21.035
Cohere	99.045
Copilot	28.005
Gemini	15.825
Grand Total	**43.586**

If you take a look at Tables 1-3, accuracy for determining if a sample of text was from Cohere was among the highest if not the highest among every model. Given the large disparity, it is a statistically significant difference and can be confidently traced back to the data, especially for the LSTM and Logistic Regression methods.

Overfitting Concerns

BERT in particular is known to struggle with overfitting smaller datasets while LSTMs usually perform better in smaller datasets [1]. While BERT did not perform excellently, it still outpaced our LSTM. Given the research on this topic, it stands to reason that our LSTM could be improved. Given the disparity between the training accuracy and test accuracy, our LSTM overfitted extremely strongly despite attempts to prevent it (Dropout layers). We did experiment with heavier dropouts, but this tended to reduce performance even further. Hyperparameters were adjusted in many ways but seemed to negatively impact performance most of the time. Given the current layer setup, this was the best result we achieved.

Discussion of Results

In the logistic regression, the model was able to guess Gemini and Cohere correctly more than any of the other models. Given the dataset, it seems likely that the model strongly overfitted on length. Ignoring that result, logistic regression performed very poorly likely because a TF-IDF's specialty does not lie in finding the contextual differences that would be required for this task.

Our LSTM model was only able to achieve an accuracy of over 75% for one model Cohere. For the rest of the models, the accuracy hovered around 50%, which is an improvement over random guessing. The model was able to gain some understanding of what differences the LLMs had but nothing concrete enough for strong performance. With a more significant dataset, we believe that the performance will improve quite a bit as this dataset may just be too small for the model to be properly trained.

Perhaps the most surprising result is Cohere not being the easiest model to predict for Bert. Instead, Copilot was the easiest for Bert to handle. This is very surprising because as a human, Copilot's responses were almost indistinguishable from Claude and Gemini's. Bert may have been able to pick up on some structural differences and found that Copilot was very uniform.

Related Work

Research on this topic has been conducted before as recently as February of this year. In [2], Rosenfield and Lazebnik attempt to analyze the differences between the stated models. These differences include Part-of-Speech distributions, dependency distributions, and text sentiment. They derived this data from the Human ChatGPT Comparison Corpus, a database containing large amounts of ChatGPT-3.5 responses to prompts regarding various fields. This was extended to include responses from ChatGPT-4o and Bard. These prompts were derived from "Finance, Medicine, Long-Form Question Answering, Open-Domain Question Answering, and Computer Science". They collected 5000 responses per model. With this data, they used an off-the-shelf XGboost model to obtain an accuracy of 88% in attributing text samples to their given models. This is highly impressive as it can determine the difference between two stages of the same model.

In [3], Chen et al use a text-to-text transformer to predict the subsequent token. This allows for the prediction to slowly and continuously be updated, implicitly predicting which LLM a model is derived from. This method performs better than BERT-based models for this specific task. They trained this model on over 340,000 text samples from the OpenLLM text dataset. The LLMs used for this experiment were GPT-3.5, PaLM, LLaMA, and GPT-2. The resulting model had an accuracy of 95.6% when it came to determining the source of any sample text. The model also had the functionality to determine whether a sample of text was human-generated as compared with any of the LLMs with accuracies of over 90%.

Citations

[1] A. Ezen-Can, "A comparison of LSTM and BERT for small corpus," arXiv:2009.05451, 2020. [Online]. Available: https://arxiv.org/abs/2009.05451

[2] A. Rosenfeld and T. Lazebnik, "Whose LLM is it Anyway? Linguistic Comparison and LLM Attribution for GPT-3.5, GPT-4 and Bard," *arXiv*, 2024. [Online]. Available: https://arxiv.org/abs/2402.14533

[3] Y. Chen, H. Kang, V. Zhai, L. Li, R. Singh, and B. Raj, "Token prediction as implicit classification to identify LLM-generated text," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2023, pp. 13112–13120. doi: 10.18653/v1/2023.emnlp-main.810.