584 Midterm Project Report

Group members: Tianrun Yu, Ming Zhu, Yucheng Zhou, Hongxi Huang

1)Abstract

For this project, we trained a deep learning classifier that can determine which Large Language Model (LLM) generated a given sentence completion. Using BERT as a baseline model, we trained and evaluated classifiers to attribute text completions generated by six different LLMs: Bart, XLNet, GPT-2, GPT-Neo, OPT, and Bloom. The dataset is curated from the Human ChatGPT Comparison Corpus (HC3), containing sentence completions from each model. Our study compares several classifiers, including XGBoost, Linear SVM, and Neural Networks, with BERT emerging as the top performer with a classification accuracy of 93%. We performed in-depth experiments to analyze how context length and linguistic features influence the classifier's performance, discovering that longer contexts improve attribution accuracy and that certain LLMs, like Bart, are easier to classify due to distinct textual features. These results show the possibility of using automated systems to identify which LLM generated a particular text.

2)Intro

As the use of LLMs grows in natural language processing tasks, figuring out which model produced a given text has become an important topic of research. Since different LLMs generate distinct outputs, even when responding to the same prompt, a key question arises: can we train a system to recognize which LLM created a specific piece of text? In this project, we focus on developing a classifier that can differentiate between sentences generated by various LLMs based on their linguistic patterns.

Our work builds upon existing research on LLM attribution, using data from six well-known LLMs: Bart, XLNet, GPT-2, GPT-Neo, OPT, and Bloom. For each of these models, we generated sentence completions from partial sentences, creating a labeled dataset. This dataset allowed us to train a classifier to predict which LLM was responsible for producing a given sentence. To do this, we looked at various linguistic and semantic characteristics, such as part-of-speech tagging, sentiment, and word similarities, to help tell the models apart.

This project contributes to the research on identifying which LLM generated a given text by evaluating the effectiveness of various classification techniques. We analyzed both traditional machine learning methods and more advanced deep learning models to understand how well they perform. Our results show that it is possible to accurately match LLM-generated text to the

model that created it, offering a useful resource for researchers and professionals working with AI-generated text.

3)Related Work

Traditionally, authorship attribution focused on identifying human authors, but with the rise of LLMs, the focus has shifted to LLM authorship attribution. As Uchendu et al. (2023) discussed, this field now addresses the challenges of attributing texts generated by different LLMs, where texts from different LLM are stylistically similar but contain subtle differences that can be used for attribution.

Research in this area has explored various methods to solve the problem of classifying AI-generated texts. For example, Abburi et al. (2023) proposed that using ensemble neural models, which combine multiple pre-trained LLMs for improved classification, is relevant to this project's goal of building a deep learning classifier to attribute text completions to their source LLM.

Another approach is to use stylometric analysis, which involves identifying unique linguistic features of different texts. Kumarage et al. (2023) introduced the use of stylometric features such as vocabulary, syntax, and structure to distinguish between the outputs of different LLM. Rosenfeld and Lazebnik (2024) also found that different LLM, such as GPT-3.5, GPT-4, and Bard, exhibit different linguistic patterns that can be used to accurately attribute texts to their source. Their findings suggest that analyzing linguistic traits is one of the strategies to improve LLM attribution.

In this project, we use BERT (Bidirectional Encoder Representations from Transformers) as our baseline model to classify the source of text completions generated by different LLMs. As introduced by Devlin et al. (2019), BERT is a pre-trained model that captures both left and right context in each layer

In addition to the core studies, other research also provides useful perspectives on LLM attribution. Li et al. (2023) reviewed how LLMs can give citations or references for their outputs to improve factual accuracy, which is relevant to model attribution. Lee et al. (2024) introduced LLM Attributor, a tool that visually shows which training data contributed to an LLM's output, helping users understand how models generate text. Lastly, Sarvazyan et al. (2023) discussed the AuTexTification shared task, which involves detecting and attributing AI-generated text across various domains, and is also relevant to improving classification accuracy.

4)Data Curation

In this experiment, we aimed to create a dataset by processing random sentences from the Human ChatGPT Comparison Corpus (HC3). We first applied shuffle cleaning to the HC3 dataset to randomize the sentence order, ensuring that any patterns or biases in the sequence were eliminated. After shuffling, we randomly selected 1000 sentences for further processing.

For each selected sentence, we extracted the first three words to form a prompt. The format of this prompt was: "Complete the sentence: [first three words of the sentence]." Here, xi represents the first three words of the sentence, and xj represents the continuation of the sentence generated by a large language model based on the prompt "Complete the sentence: xi."

We used several large language models, including Bart, XLNet, GPT-2, GPT-Neo, OPT, and Bloom, to generate completions for each prompt. For each model, we generated 1000 samples, resulting in a total of 6000 generated sentences. The label for each sample corresponds to the model used for the generation. Therefore, we obtained our desired balanced dataset with 3 columns: xi, xj, LLM.

This method allowed us to create a diverse dataset consisting of sentence completions from six different models, which will be used for further evaluation and comparison in our analysis.

5)Baseline Models, Experiments and Results

5.1 Feature Extraction Methodology:

We have explored four different classifiers in our project. In the first three experiments, we applied the same feature extraction process to generate a set of features that captured the linguistic, syntactic, and semantic characteristics of the sentences. These features were designed to differentiate between sentences generated by different large language models (LLMs). The following features were extracted from each sentence pair (xi and xj):

1. **Vocabulary Size**: The number of unique tokens in the sentence.
2. **POS Tags Distribution**: The proportion of different part-of-speech (POS) tags in the sentence.
3. **Syntactic Dependencies**: The distribution of syntactic dependencies in the sentence.
4. **Sentiment Score**: The sentiment polarity score of the sentence, ranging from -1 (negative) to 1 (positive), using the NLTK SentimentIntensityAnalyzer.
5. **Cosine Similarity**: The cosine similarity between xi (the prompt) and xj (the generated continuation), indicating the similarity between the original and generated text.

These features were combined into a feature matrix that was used to train classifiers in our experiments.

5.2 Experiment 1: XGBoost Classifier

For the first experiment, we used the XGBoost classifier to predict which LLM generated each sentence. XGBoost was selected due to its ability to handle large feature sets and its effectiveness in structured data tasks.

Training and Testing:

The dataset was split into 80% training data and 20% test data. We trained the XGBoost model using default hyperparameters, and then evaluated the model on the test set.

Evaluation Metrics:

In this project, we used a variety of evaluation metrics to assess the performance of the LLM attribution classifiers. The primary metric was accuracy, which measures the proportion of correctly classified text completions out of all samples. To better understand the classifier's performance across different LLM classes, we also calculated the precision, recall, and F1-score for each model individually. Precision quantifies the proportion of true positives among all predicted positives, indicating how many of the texts attributed to an LLM are correctly classified. Recall measures the proportion of true positives identified out of all actual positives, reflecting the ability of the classifier to capture all instances of a particular LLM. The F1-score, which is the harmonic mean of precision and recall, was used to provide a balanced view of the classifier’s performance, particularly when dealing with any potential class imbalances. Additionally, we calculated macro-averaged metrics to evaluate the overall performance across all LLMs equally, giving a more holistic view of the classifier’s ability to differentiate between the six LLMs in the dataset. These metrics were used to provide detailed insights into model performance and to guide improvements throughout the experiments.

Results:

	Precision	**Recall**	**F1-Score**	**Support**
Bart	1.00	1.00	1.00	197
Bloom	0.92	0.97	0.94	201
GPT-2	0.74	0.76	0.75	198
GPT-Neo	0.75	0.70	0.73	192
OPT	0.96	0.94	0.95	211

XLNet	0.95	0.97	0.96	201
Accuracy	-	-	0.89	1200
Macro Avg	0.89	0.89	0.89	1200
Weighted avg	0.89	0.89	0.89	1200

5.3 Experiment 2: Linear Support Vector Machine Classifier

For the second experiment, we used a Linear Support Vector Machine (SVM) classifier to predict which LLM generated each sentence. SVM was chosen for its ability to sometimes provide more accurate predictions in cases where the model is simpler but the features capture key characteristics of the data.

Training and Testing:

The same features, training and testing data split (80% training, 20% testing), and evaluation metrics from the XGBoost experiment were used here. We trained the SVM using default hyperparameters.

Results:

	Precision	**Recall**	**F1-Score**	**Support**
Bart	0.98	1.00	0.99	197
Bloom	0.77	0.90	0.83	201
GPT-2	0.50	0.64	0.56	198
GPT-Neo	0.48	0.39	0.43	192
OPT	0.84	0.80	0.82	211
XLNet	0.72	0.57	0.64	201
Accuracy	-	-	0.72	1200
Macro Avg	0.72	0.72	0.71	1200

Weighted avg	0.72	0.72	0.71	1200

5.4 Experiment 3: Neural Network Classifier

In the third experiment, we employed a **neural network classifier** using PyTorch. The input features were the same as in Experiment 1: vocabulary size, POS tags, syntactic dependencies, sentiment scores, and cosine similarities. Unlike in Experiment 1, we used a neural network instead of XGBoost to classify the sentences generated by different LLMs.

Techniques Used:

- **Feature Scaling**: Before feeding the features into the neural network, we performed **standardization** on all features. This ensured that each feature had a mean of 0 and a standard deviation of 1, helping the neural network converge faster during training and improving overall performance.
- **Regularization**: To prevent overfitting, we incorporated two main regularization techniques:
    - **Dropout**: After each hidden layer, a **dropout layer** with a dropout rate of 0.5 was added. This randomly "dropped out" 50% of neurons during each training step to prevent the model from overfitting to the training data.
    - **Batch Normalization**: After each fully connected layer, we applied **batch normalization**. This normalized the output of the layer, helping to stabilize learning by reducing internal covariate shift and speeding up convergence.

Neural Network Architecture:

- **Input Layer**: The input layer consisted of all manually extracted features.
- **Hidden Layers**:
    - **First Hidden Layer**: This layer had 256 neurons, followed by batch normalization to stabilize learning, ReLU activation for non-linearity, and **dropout (rate=0.5)** to prevent overfitting.
    - **Second Hidden Layer**: This layer had 128 neurons, again followed by batch normalization, ReLU activation, and **dropout (rate=0.5)**.
- **Output Layer**: A fully connected layer with neurons equal to the number of classes (i.e., the number of LLMs) to produce the final classification output using a softmax activation function.

Training and Testing:

As in Experiment 1, the dataset was split into 80% training and 20% testing data. The neural network was trained using the Adam optimizer with cross-entropy loss over 100 epochs.

Results:

	Precision	Recall	F1-Score	Support
Bart	0.99	1.00	1.00	197
Bloom	0.79	0.92	0.85	201
GPT-2	0.83	0.56	0.66	198
GPT-Neo	0.63	0.83	0.72	192
OPT	0.61	0.80	0.85	211
XLNet	0.98	0.96	0.97	201
Accuracy	-	-	0.84	1200
Macro Avg	0.86	0.84	0.84	1200
Weighted avg	0.86	0.84	0.84	1200

5.5 Experiment 4: BERT Model

For this experiment, we used a BERT (Bidirectional Encoder Representations from Transformers) model to classify which large language model (LLM) generated each sentence. BERT was selected due to its pre-training on large text corpora, making it effective at capturing both left and right context in each layer of the model.

Training and Testing:

We fine-tuned the BERT model as a classifier by concatenating (xi, xj) as a single sequence with a separator token `[SEP]`. We used the “bert-base-uncased” version from the Huggingface Transformers library, with a `max_length` set to 128 tokens. The batch size was set to 16, the weight decay factor was set to 0.01, and the model used 500 warmup steps.

After tokenizing the texts using the model’s tokenizer, we trained the BERT model for 10 epochs. The model converged after about 20 minutes on a T4 GPU in Google Colab. We evaluated the model on the test dataset after training.

Results:

	Precision	Recall	F1-Score	Support

Bart	1.00	1.00	1.00	197
Bloom	0.91	0.95	0.93	201
GPT-2	0.89	0.84	0.86	198
GPT-Neo	0.84	0.87	0.85	192
OPT	0.94	0.92	0.93	211
XLNet	0.99	1.00	0.99	201
Accuracy	-	-	0.93	1200
Macro Avg	0.93	0.93	0.93	1200
Weighted avg	0.93	0.93	0.93	1200

We can observe that BERT achieved an overall classification accuracy of 0.93, which outperforms all previous methods we have tried. It also performed well on Bart, XLNet, and OPT data. The scores on Bloom, GPT-2, and GPT-Neo are slightly lower, but are also satisfactory. Therefore, we finalize this BERT as our best optimized baseline.

6)In-depth Experiments & Analysis

By using our optimized BERT classifier model as baseline, we designed two in-depth experiments to further analyze the feature importance and model behavior.

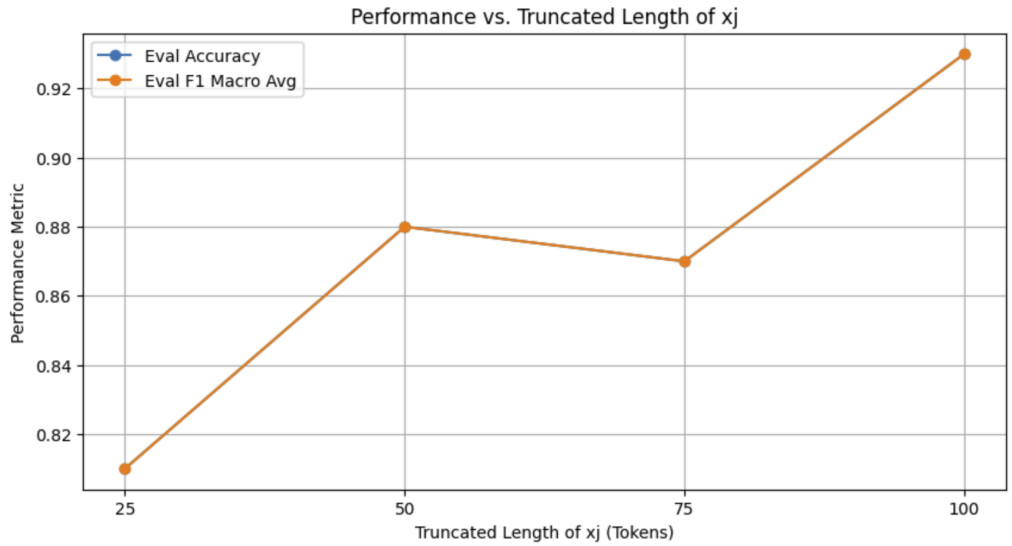
1.Context Length Influence Analysis
2.Feature Analysis for GPT-2 and GPT-Neo vs. Bart

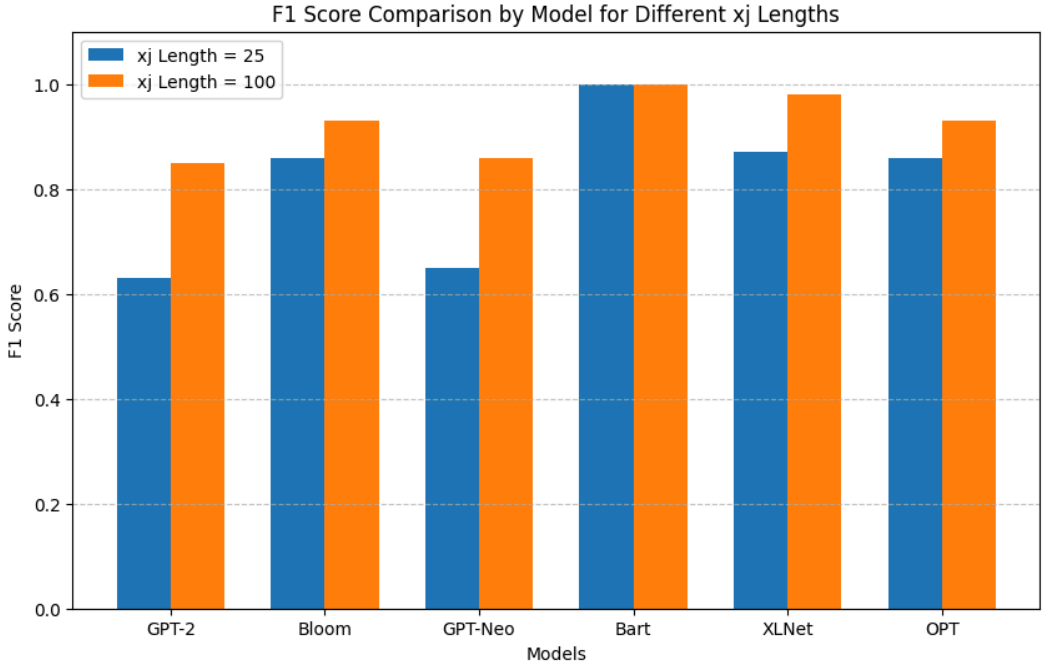
Experiment 1. Context Length Influence

For the first experiment, we aimed to analyze how the length of xj affects the classifier's performance. Does longer context lead to better attribution, or do certain LLMs behave more distinctively with shorter prompts? Therefore, in this experiment, we systematically varied the lengths of xj and observed how it affects the classifier's performance.

About the experiment setting, we used the same processed training and testing labeled dataset containing (xi, xj, LLM) pairs as previous experiments to create multiple context-length datasets. The baseline model used here is BERT for sequence classification, fine-tuned on model attribution task. And we used metrics like accuracy, F1-score, precision, and recall for performance evaluation.

First, we prepared multiple context-length datasets. We tokenized xj and truncated them to different lengths like [25, 50, 75, 100] tokens. Then we created several versions of our training and test datasets, each with a different length for xj. Next, we trained and evaluated BERT on each dataset and tracked performance metrics for each model trained on a different context length.





By observing the results, we can clearly notice that longer contexts lead to better attribution accuracy. In addition, Bart, Bloom, XLNet, and OPT have relatively high f1-score even when the context lengths are short(like xj length = 25), especially Bart(1.00). This means that the context length might have less influence on these models' classification. In contrast, as the context length becomes longer, the f1-score for GPT-2 and GPT-Neo significantly increases. For example, the f1-score for GPT-2 when xj length = 25 is 0.63, while f1-score for GPT-2 when xj length = 100 is 0.85. These results demonstrate that context length has a large impact on the classification performance of GPT-2 and GPT-Neo.

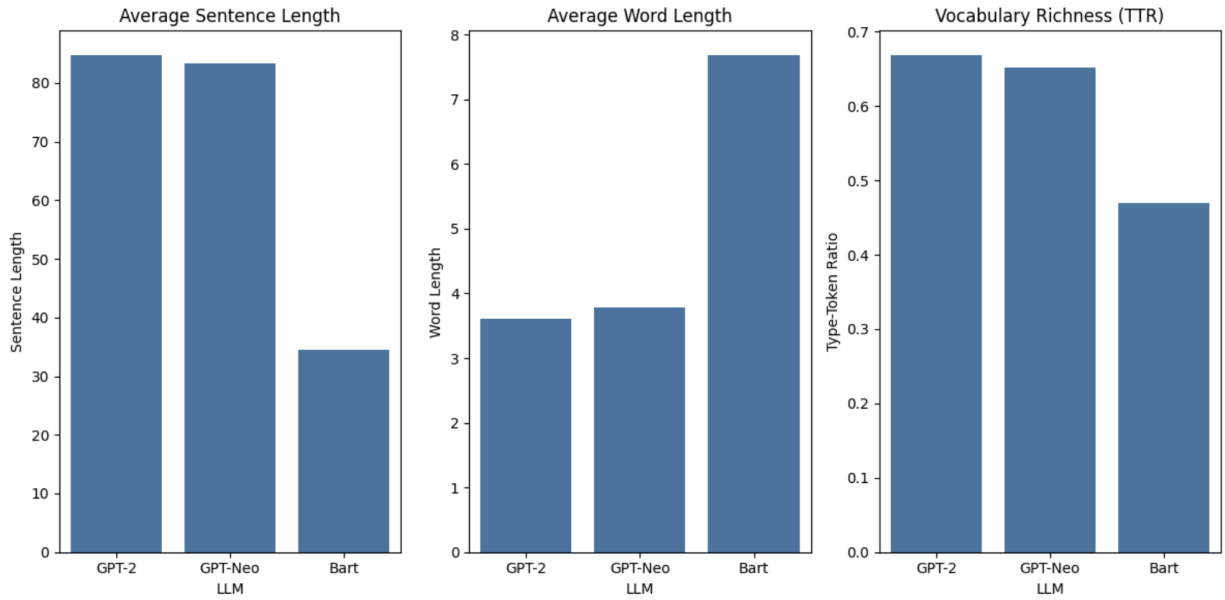
Experiment 2. Feature Analysis

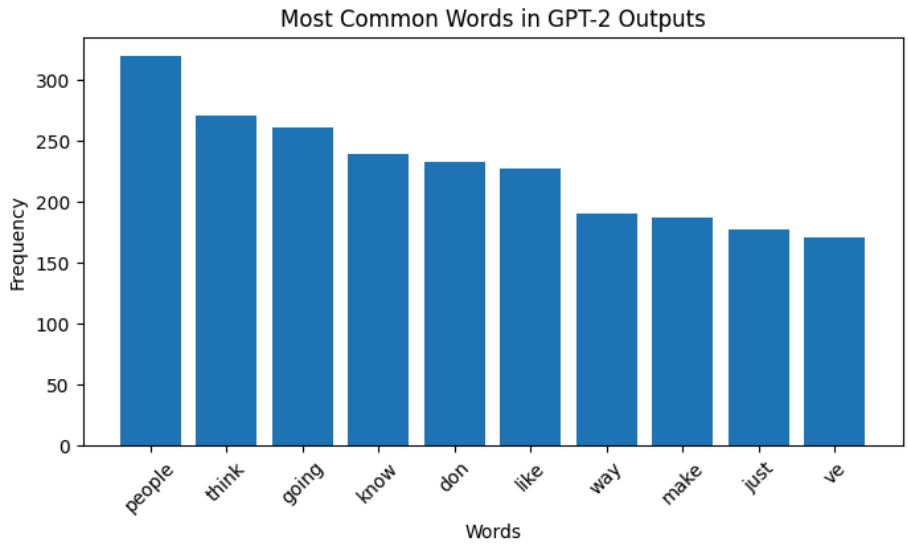
In the second experiment, we wanted to analyze the special behavior of GPT-2 and GPT-Neo vs. Bart. We doubted that the BERT classifier struggles to distinguish text generated by GPT-2 and GPT-Neo because they produce more similar patterns, either to each other or to other LLMs, compared to Bart. Therefore, we performed feature analysis to see how each model's generated text compares on features like word frequency, sentence length, vocabulary richness, and stylistic markers.

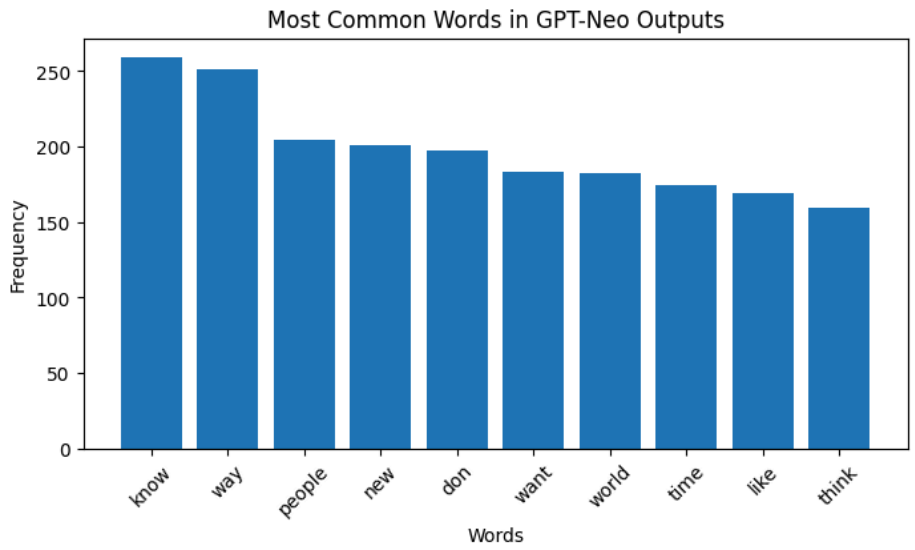
We first loaded the train and test datasets into pandas DataFrames. Then we filtered out rows corresponding to GPT-2, GPT-Neo, and Bart. And we computed various features for each LLM-generated text.

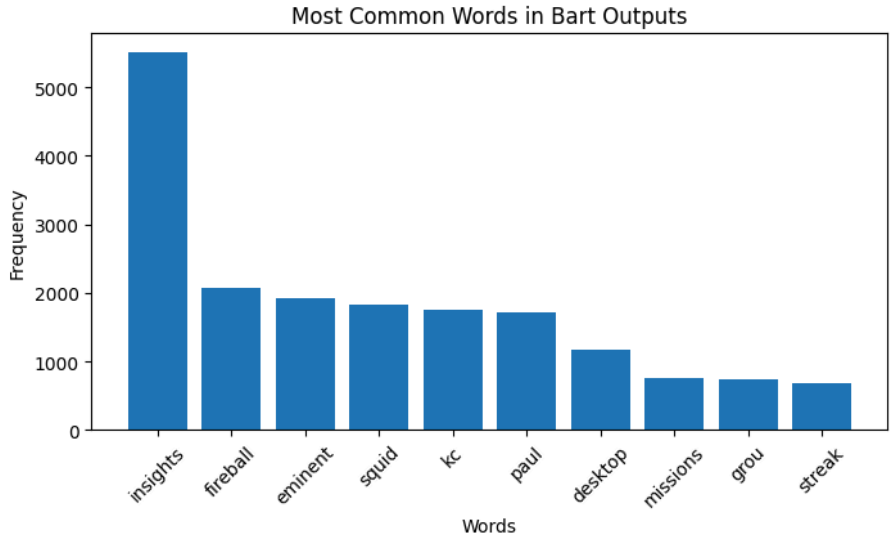
Key Features we Analyzed:
-Sentence Length: The average number of tokens in the generated text.
-Average Word Length: The mean length of words in the generated text.
-Vocabulary Richness (Type-Token Ratio): The ratio of unique words to the total number of words in the generated text.
-Top 10 Most Common Words: The most frequently occurring words in each LLM's output, excluding stop-words.

Visualizations:


Average Sentence Length
Sentence Length
0
10
20
30
40
50
60
70
80
GPT-2
GPT-Neo
Bart
LLM
Average Word Length
Word Length
0
1
2
3
4
5
6
7
8
GPT-2
GPT-Neo
Bart
LLM
Vocabulary Richness (TTR)
Type-Token Ratio
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
GPT-2
GPT-Neo
Bart
LLM


Most Common Words in GPT-2 Outputs
Frequency
0
50
100
150
200
250
300
people
think
going
know
don
like
way
make
just
ve
Words





For average sentence length and average word length graphs, we can see that GPT-2 and GPT-Neo perform similarly, while Bart performs largely differently than those two. The generated context produced by Bart tends to have relatively shorter average sentence length but longer average word length compared to GPT-2 and GPT-Neo generated context. The Vocabulary Richness (TTR) graph shows that GPT-2 and GPT-Neo tend to have more varied vocabulary usage. In addition, GPT-2 and GPT-Neo have many overlaps on the most common words in outputs like "people", "think", and "way", etc. In contrast, there's no overlap on the most common words between Bart and GPT models. The most common word for Bart model is "insights" which has a frequency of over 5,000. This number is much higher than the GPTs most common words' frequency(around 200~300). Thus, this most_common_words feature might be a potential reason for the high f1-score and classification performance for Bart model.

7)Conclusion and Future Work

In this project, we explored the task of LLM attribution, aiming to classify the text completions generated by various Large Language Models (LLMs). We created a dataset consisting of 6000 generated sentences by extracting prompts from the Human ChatGPT Comparison Corpus (HC3) and using six different LLMs (Bart, XLNet, GPT-2, GPT-Neo, OPT, and Bloom) to produce continuations for truncated sentences. Our study demonstrates that it is possible to identify the source LLM of a given piece of text using a classifier, particularly by leveraging BERT as a sequence classification model. We tested the performance of various machine learning and deep learning approaches, including XGBoost, Linear SVM, neural networks, and BERT, finding that BERT achieved the best overall performance with an accuracy of 93%.

The detailed experiments revealed interesting insights into how different LLMs generate text. Notably, the classifier was highly effective in identifying outputs from Bart, XLNet, and OPT, achieving near-perfect F1-scores. However, the classifier faced more challenges when working with GPT-2 and GPT-Neo, indicating that these models produce stylistically similar text that makes it harder to distinguish. Our in-depth analysis of context length showed that longer contexts led to improved classification performance, particularly for GPT-2 and GPT-Neo, where F1-scores significantly improved with increased context length. Additionally, feature analysis of generated text revealed that Bart's distinct stylistic markers and vocabulary contributed to its easier classification.

These findings contribute to the understanding of LLM behaviors, how text generated by different models can be effectively distinguished, and what features contribute most to successful attribution. This work has applications in fields like content generation, AI safety, and transparency, where knowing the source of AI-generated text is important.

Several directions can be pursued to extend this work. For instance, to try more sophisticated attribution models. Extend this study to incorporate more advanced LLM classifiers beyond BERT, such as RoBERTa, ALBERT, or transformer models with larger capacities like GPT-3, to explore how their deeper architectures and improved context understanding affect model attribution accuracy. Moreover, researchers can explore how context domain (e.g., news, conversational text, academic content) and stylistic changes impact classification performance. This will help understand whether certain LLMs are better suited for specific domains and if domain-based context influences model attribution.

In summary, this project lays a foundation for effective LLM attribution using deep learning techniques, particularly with BERT as a strong baseline.

Reference

Abburi, H., Suesserman, M., Pudota, N., Veeramani, B., Bowen, E., & Bhattacharya, S. (2023). Generative ai text classification using ensemble llm approaches. *arXiv preprint arXiv:2309.07755*.

Devlin, J. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Gagiano, R., Fayek, H., Kim, M. M. H., Biggs, J., & Zhang, X. (2023). IberLEF 2023 AuTexTification: Automated Text Identification Shared Task-Team OD-21. In *IberLEF@ SEPLN*.

Kumarage, T., & Liu, H. (2023, November). Neural Authorship Attribution: Stylometric Analysis on Large Language Models. In *2023 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)* (pp. 51-54). IEEE.

Lee, S., Wang, Z. J., Chakravarthy, A., Helbling, A., Peng, S., Phute, M., ... & Kahng, M. (2024). LLM Attributor: Interactive Visual Attribution for LLM Generation. *arXiv preprint arXiv:2404.01361*.

Li, D., Sun, Z., Hu, X., Liu, Z., Chen, Z., Hu, B., ... & Zhang, M. (2023). A survey of large language models attribution. arXiv preprint *arXiv:2311.03731*.

Rosenfeld, A., & Lazebnik, T. (2024). Whose LLM is it Anyway? Linguistic Comparison and LLM Attribution for GPT-3.5, GPT-4 and Bard. *arXiv e-prints, arXiv-2402*.

Sarvazyan, A. M., González, J. Á., Franco-Salvador, M., Rangel, F., Chulvi, B., & Rosso, P. (2023). Overview of autextification at iberlef 2023: Detection and attribution of machine-generated text in multiple domains. *arXiv preprint arXiv:2309.11285*.

Uchendu, A., Le, T., & Lee, D. (2023). Attribution and obfuscation of neural text authorship: A data mining perspective. *ACM SIGKDD Explorations Newsletter, 2*5(1), 1-18.