Midterm Project

Group:

Doyle Yeh, Eric Chou, Guodong Dong, Asfahan Shah

Pennsylvania State University

CSE 584: Machine Learning: Tools and Algorithms

Prof. Wenpeng Yin

October 6, 2024

Content

1. Introduction 1

2. Dataset 1

2.1 Text extraction 2

2.2 Sentence generation 2

2.3 Dataset creation 3

3. Classifier 4

3.1 Training 4

4. Result 6

4.1 Conclusion 11

5. Analysis / experiments 13

5.1 Model Performance Comparisons 13

5.2 Truncated text generated model 17

6. Related work 19

6.1 Model attribution and detection 19

6.2 Text classification and representation 20

6.3 Benchmarking and evaluation of LLMs 21

6.4 Natural language generation (NLG) evaluation 22

References 23

1. Introduction

In the field of Natural Language Processing (NLP), Large Language Models (LLMs) have significantly advanced the generation of coherent and contextually relevant text. A critical challenge within this domain is the identification of the specific LLM responsible for generating a particular completion in response to a given text prompt. This project tries to develop a deep learning classifier that can accurately ascertain which LLM produced the completed texts. This work attempts to improve our understanding of LLM behaviors and capabilities by identifying the differences between various models' text creation processes. These kinds of understandings are critical to improving and optimizing LLM performance in a variety of NLP tasks, such as information retrieval, content creation, and other applications where text generation is critical.

In this project, we aim to build a deep learning classifier that can identify which Large Language Model (LLM) generated a specific text completion. The process begins with a truncated text fragment (x_i), such as "Yesterday I went," which is written by a human. Then we complete the sentence by different LLMs which produce varying continuations (x_j) such as "to Costco and purchased a floor cleaner." The classifier will be trained to recognize patterns in the completed text pairs (x_i, x_j) and classify which LLM produced the completion based on stylistic, linguistic, or structural differences inherent to each model.

2. Dataset

We are provided with datasets from five prominent LLMs: GPT-2, GPT-4, GPT-NEO, Gemini, and Reformer, each containing 5000 completed sentences. This yields a comprehensive dataset of 25,000 text pairs consisting of truncated texts (x_i) and their corresponding completions (x_j). The goal is to train a classifier that can accurately attribute each text pair to the correct LLM based on their distinctive text generation characteristics.

For the manually written content, we use BookCorpus which is a large collection of books used for training and evaluating natural language processing models. It contains over 11,000 books from various genres, primarily unpublished novels, and is widely used in pre-training language models like BERT and GPT. Since the data was from books, we can make sure that the content is written by humans.

For the data processing, there are three steps in total which are text extraction, sentence generation, and training dataset creation.

2.1 Text extraction

In this step, we first imported and shuffled the BookCorpus dataset to introduce randomness. From the shuffled data, we selected 5,000 samples that met our specified minimum length requirements. To generate incomplete sentences, we truncated each sample from the middle. For samples that exceeded a certain length, we extracted only the first set number of words. This process ensured a consistent approach to creating incomplete text fragments. As a result, we constructed a dataset with a column labeled x_i, containing 5,000 truncated sentence samples, ready for subsequent processing.

2.2 Sentence generation

In the sentence generation process, we utilized the Hugging Face Transformers library to call various large language models (LLMs) and their tokenizers for completing the truncated sentences. The models included GPT-2, GPT-Neo, XLNet, BART, T5, Pegasus, and Reformer. However, many of these models struggled to produce coherent sentence completions, often generating nonsensical, repetitive, or null outputs, particularly with XLNet, BART, T5, and Pegasus. As a result, we opted to manually generate sentence completions using GPT-4 and Gemini to ensure better quality and meaningful comparisons. Once the completions (x_j) were obtained, we appended them to the dataset, pairing them with their corresponding truncated sentences (x_i) for further analysis.

2.3 Dataset creation

In the final step, we prepared five sub-datasets generated by GPT-2, GPT-4o, GPT-Neo, Reformer, and Gemini. Each sub-dataset contained two columns: x_i (truncated sentences) and x_j (their respective continuations). We then labeled each sub-dataset according to the LLM used to generate the completions and concatenated them into a single dataset. This unified dataset, containing both the input pairs and their corresponding labels, was then ready to be used for training and evaluating the classification model.

3. Classifier

For the classification task, our approach is to fine tune a large language model on our dataset. Specifically, we chose ALBERT, DistilBert and RobertA models. A brief description of each model is given below:

ALBERT: It is a type of BERT which is designed to be more efficient in terms of model size and training time while not compromising too much performance. For our experiment we are using ALBERT-Base which has about 12 million parameters.

DistilBERT: It is a smaller, faster, and lighter version of BERT developed by Hugging Face. It is developed via knowledge distillation. DistilBERT has about 66 million parameters.

RoBERTa: It is a robustly optimized version of BERT developed by Facebook AI. RoBERTa as compared to BERT, is trained on a larger dataset and with different training techniques, which improves its performance. We are using RoBERTa-Base which has about 125 million parameters.

3.1 Training

For both ALBERT and DistilBERT, we directly fine-tuned the models using AdamW optimizer and finetuned it for 3 epochs. Given RoBERTa's larger number of parameters and our computational limitations, we employed the Low-Rank Adaptation (LoRA) technique for fine-tuning.

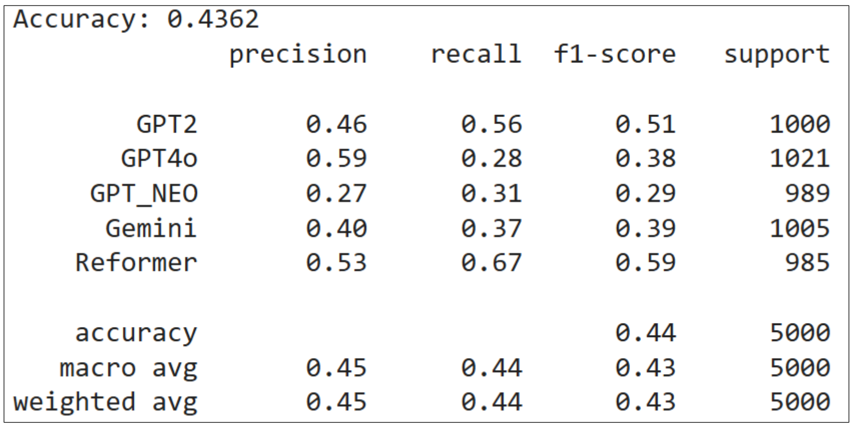
LoRA is an efficient method for fine-tuning large language models. Instead of adjusting all the training all parameters w of a layer, we freeze w and fine-tune two low-rank matrices. These low-rank matrices contain significantly fewer learnable parameters compared to w, allowing for faster and more efficient fine-tuning. We fine-tuned RoBERTa for both 3 and 10 epochs.

The train test split ratio for the experiments is about 0.2. All the other hyperparameters are kept standard and are present in the files provided in GitHub repositiory.

Although we considered using more advanced models like GPT 2 or LLaMA as the backbone for the classifier, but due to lack of computation resources and these models possessing very large number of parameters, we choose BERT based models.

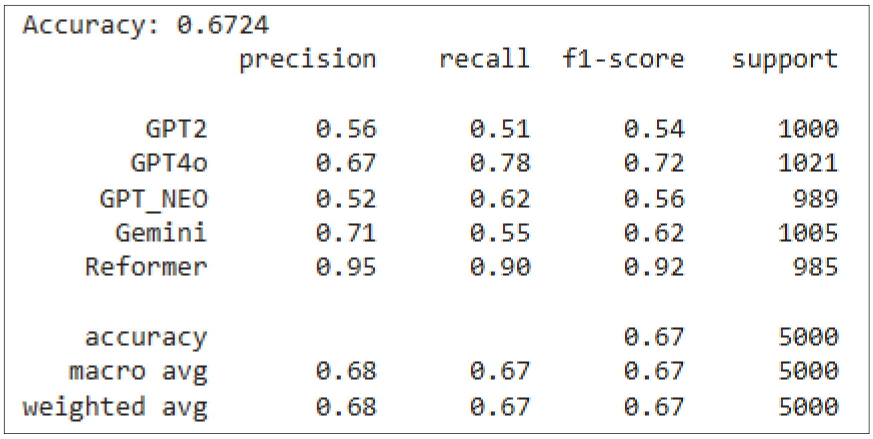
4. Result

The classification performance metrics of three models are depicted in Figure 1-2. A detailed analysis of these results will be conducted in the Analysis/Experiments section.



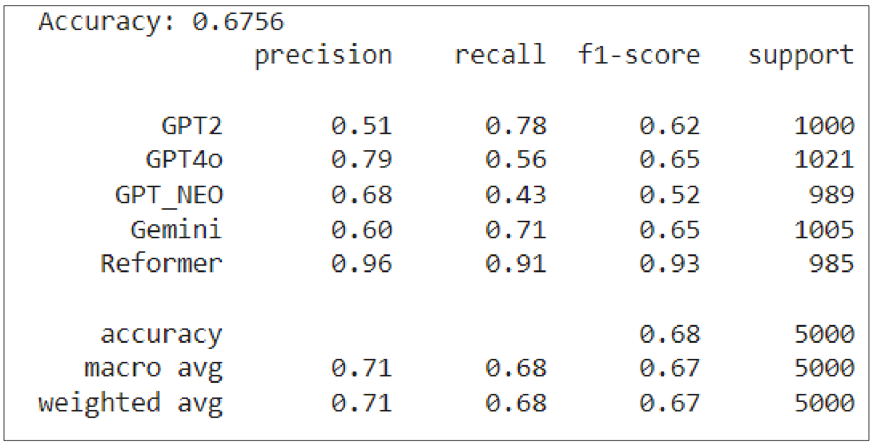
```
Accuracy: 0.4362
              precision    recall  f1-score   support

        GPT2       0.46      0.56      0.51      1000
       GPT4o       0.59      0.28      0.38      1021
     GPT_NEO       0.27      0.31      0.29       989
      Gemini       0.40      0.37      0.39      1005
    Reformer       0.53      0.67      0.59       985

    accuracy                           0.44      5000
   macro avg       0.45      0.44      0.43      5000
weighted avg       0.45      0.44      0.43      5000
```

▲ Figure 1: ALBERT Model Classification Performance Metrics



```
Accuracy: 0.6724
              precision    recall  f1-score   support

        GPT2       0.56      0.51      0.54      1000
       GPT4o       0.67      0.78      0.72      1021
     GPT_NEO       0.52      0.62      0.56       989
      Gemini       0.71      0.55      0.62      1005
    Reformer       0.95      0.90      0.92       985

    accuracy                           0.67      5000
   macro avg       0.68      0.67      0.67      5000
weighted avg       0.68      0.67      0.67      5000
```

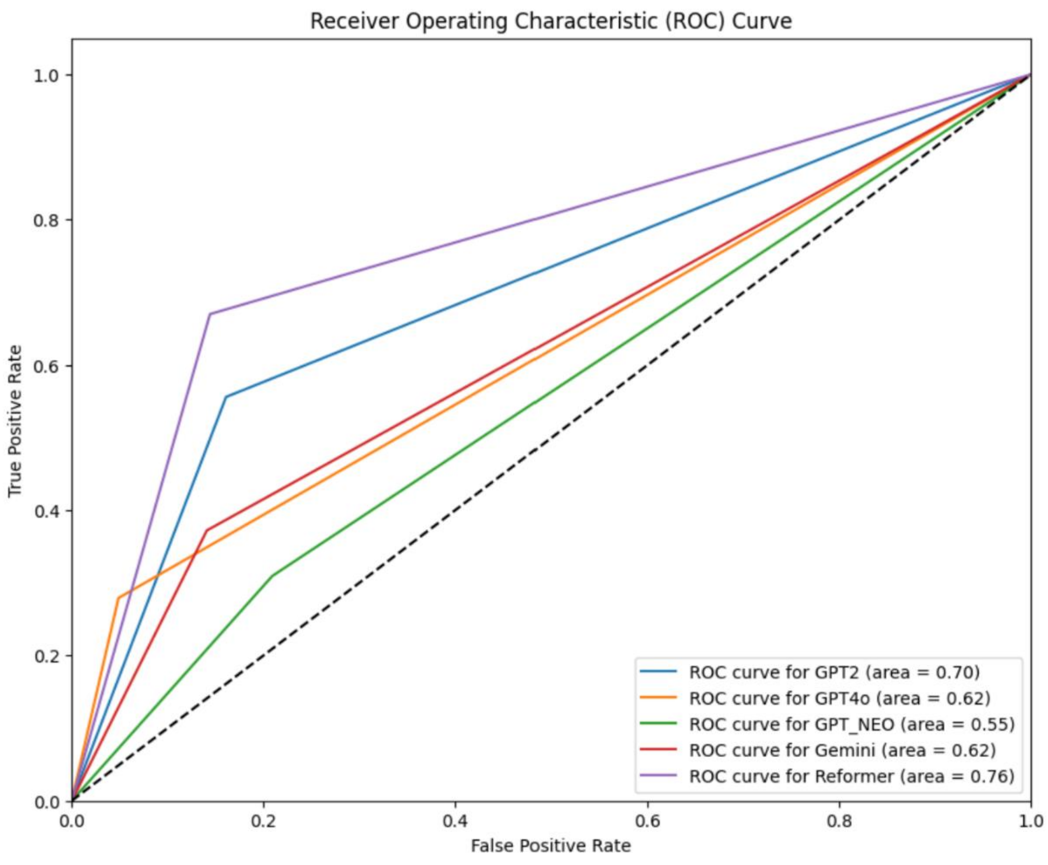
▲ Figure 2: DistilBERT Model Classification Performance Metrics



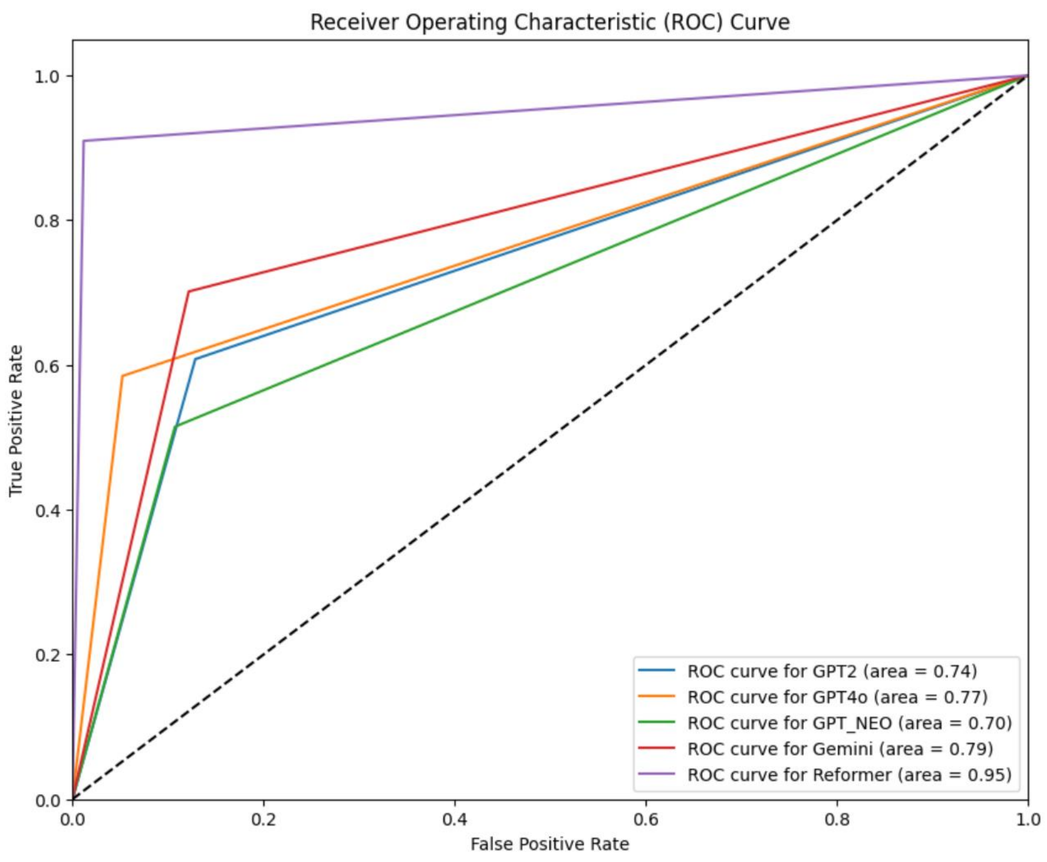
```
Accuracy: 0.6756
              precision    recall  f1-score   support

        GPT2       0.51      0.78      0.62      1000
       GPT4o       0.79      0.56      0.65      1021
     GPT_NEO       0.68      0.43      0.52       989
      Gemini       0.60      0.71      0.65      1005
    Reformer       0.96      0.91      0.93       985

    accuracy                           0.68      5000
   macro avg       0.71      0.68      0.67      5000
weighted avg       0.71      0.68      0.67      5000
```

▲ Figure 3: RoBERTa Model Classification Performance Metrics

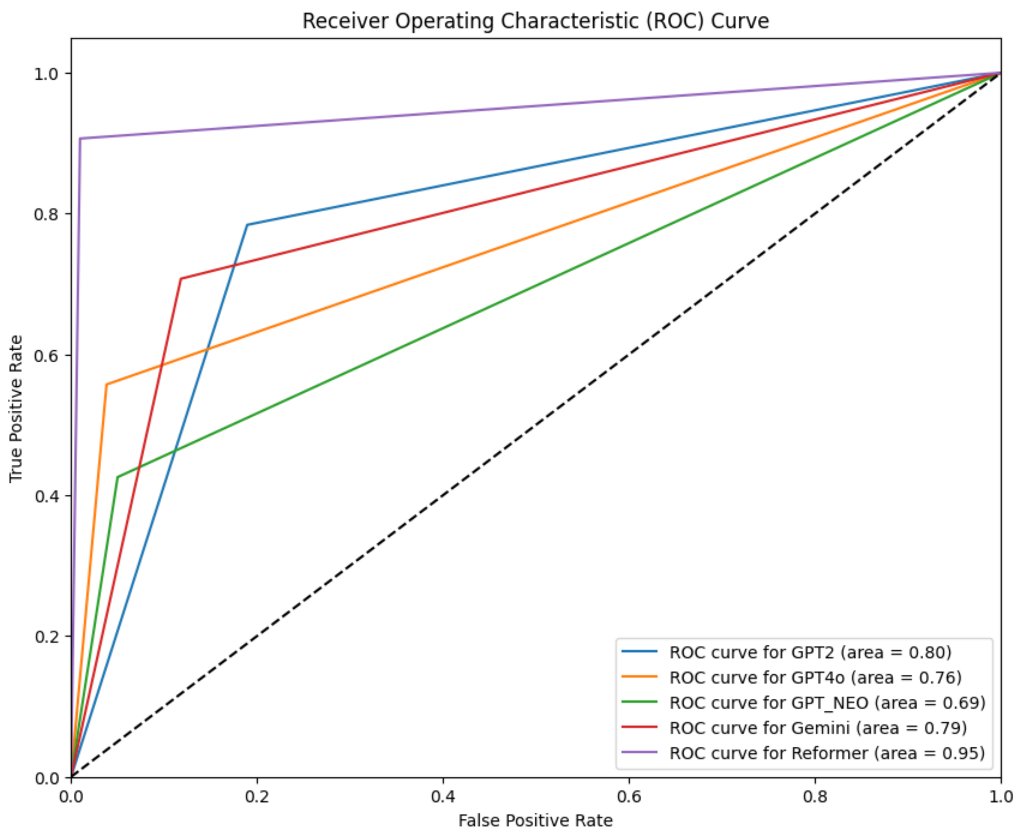
The ROC curves of three models are illustrated in Figures 4-6. A comprehensive analysis of these results will be carried out in the Analysis/Experiments section.



▲Figure 4: ALBERT Model ROC Curves

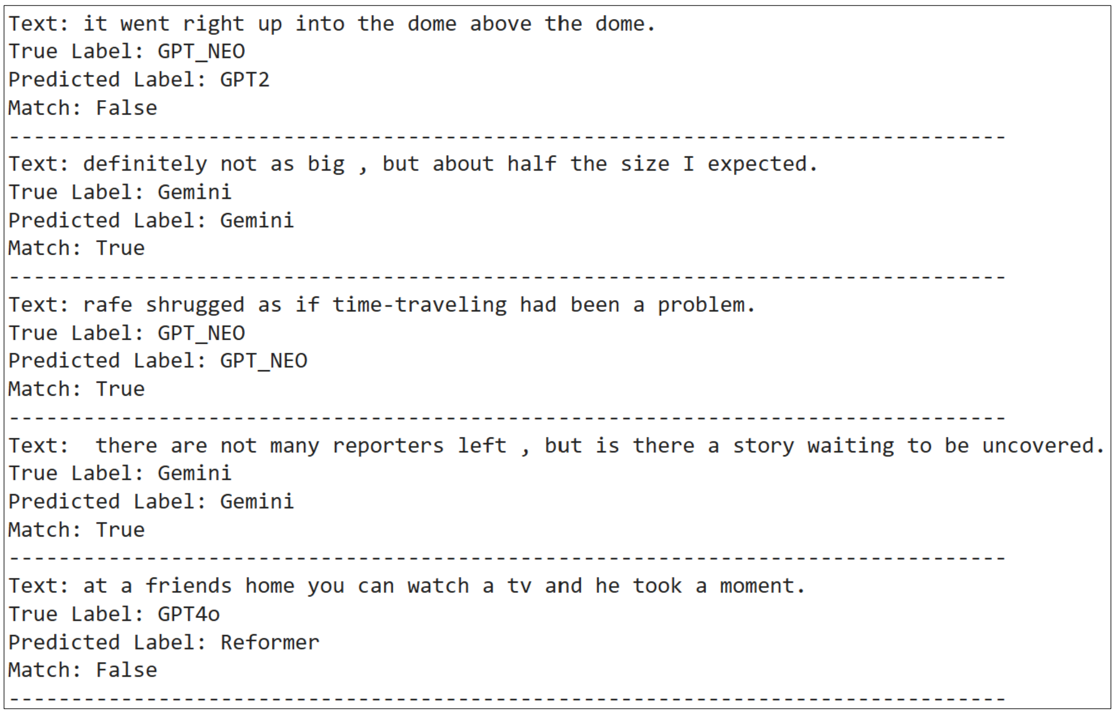


▲ Figure 5: DistilBERT Model ROC Curves



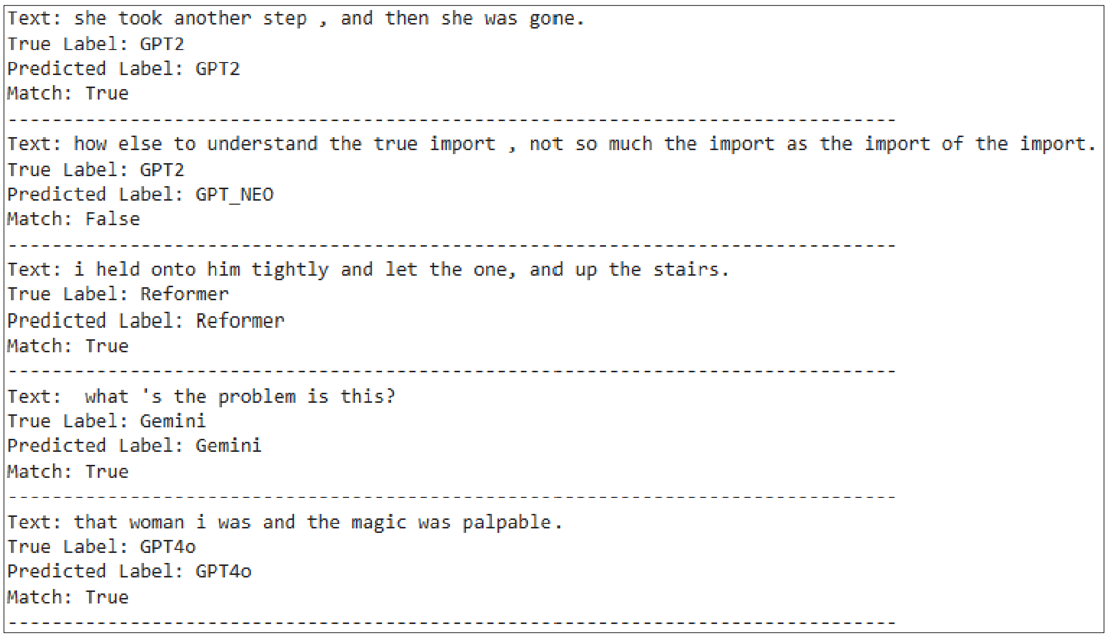
▲ Figure 6: RoBERTa Model ROC Curves

Figures 7-9 display the comparison between predictions and ground truth for the three models. A thorough analysis of these results will be conducted in the Analysis/Experiments section.



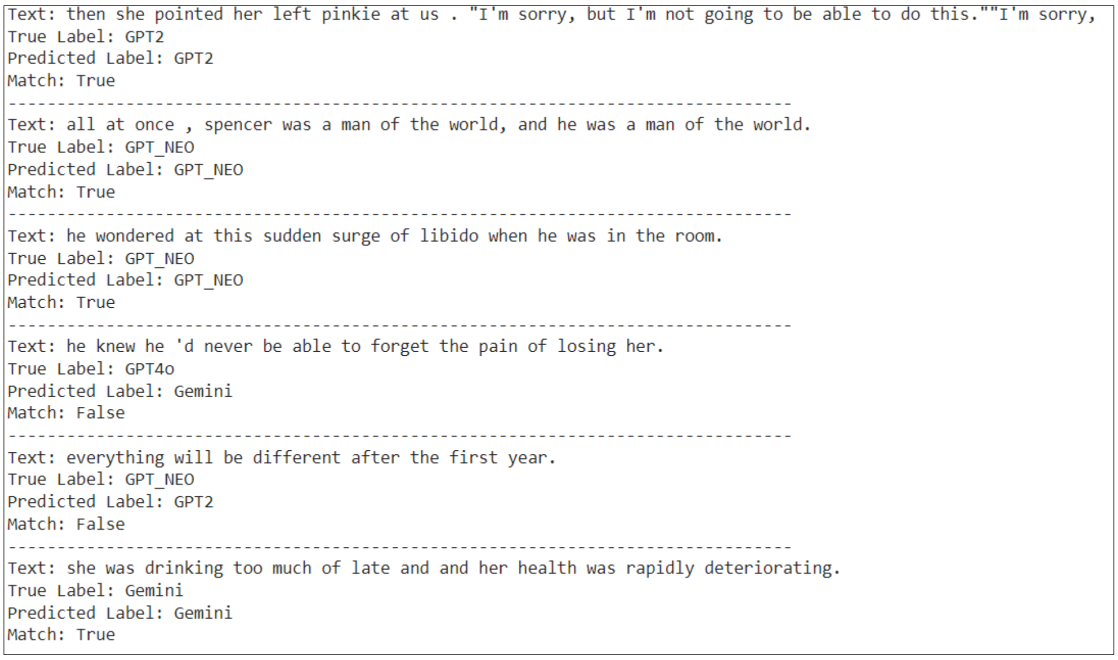
```
Text: it went right up into the dome above the dome.
True Label: GPT_NEO
Predicted Label: GPT2
Match: False
--------------------------------------------------------------------------------
Text: definitely not as big , but about half the size I expected.
True Label: Gemini
Predicted Label: Gemini
Match: True
--------------------------------------------------------------------------------
Text: rafe shrugged as if time-traveling had been a problem.
True Label: GPT_NEO
Predicted Label: GPT_NEO
Match: True
--------------------------------------------------------------------------------
Text:  there are not many reporters left , but is there a story waiting to be uncovered.
True Label: Gemini
Predicted Label: Gemini
Match: True
--------------------------------------------------------------------------------
Text: at a friends home you can watch a tv and he took a moment.
True Label: GPT4o
Predicted Label: Reformer
Match: False
--------------------------------------------------------------------------------
```

▲ Figure 7: ALBERT Model Predictions and Ground Truth Comparison



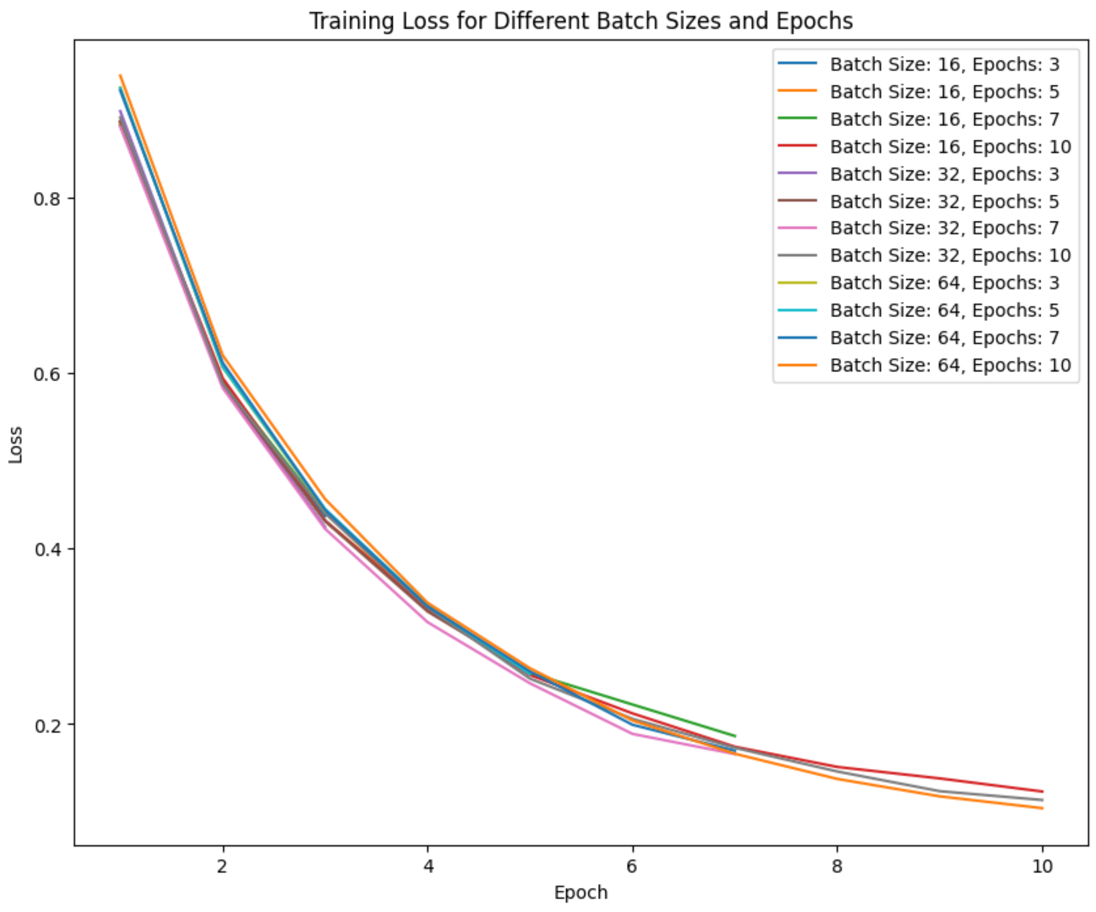
```
Text: she took another step , and then she was gone.
True Label: GPT2
Predicted Label: GPT2
Match: True
--------------------------------------------------------------------------------
Text: how else to understand the true import , not so much the import as the import of the import.
True Label: GPT2
Predicted Label: GPT_NEO
Match: False
--------------------------------------------------------------------------------
Text: i held onto him tightly and let the one, and up the stairs.
True Label: Reformer
Predicted Label: Reformer
Match: True
--------------------------------------------------------------------------------
Text:  what 's the problem is this?
True Label: Gemini
Predicted Label: Gemini
Match: True
--------------------------------------------------------------------------------
Text: that woman i was and the magic was palpable.
True Label: GPT4o
Predicted Label: GPT4o
Match: True
--------------------------------------------------------------------------------
```

▲ Figure 8: DistilBERT Model Predictions and Ground Truth Comparison



```
Text: then she pointed her left pinkie at us . "I'm sorry, but I'm not going to be able to do this.""I'm sorry,
True Label: GPT2
Predicted Label: GPT2
Match: True
--------------------------------------------------------------------------------
Text: all at once , spencer was a man of the world, and he was a man of the world.
True Label: GPT_NEO
Predicted Label: GPT_NEO
Match: True
--------------------------------------------------------------------------------
Text: he wondered at this sudden surge of libido when he was in the room.
True Label: GPT_NEO
Predicted Label: GPT_NEO
Match: True
--------------------------------------------------------------------------------
Text: he knew he 'd never be able to forget the pain of losing her.
True Label: GPT4o
Predicted Label: Gemini
Match: False
--------------------------------------------------------------------------------
Text: everything will be different after the first year.
True Label: GPT_NEO
Predicted Label: GPT2
Match: False
--------------------------------------------------------------------------------
Text: she was drinking too much of late and and her health was rapidly deteriorating.
True Label: Gemini
Predicted Label: Gemini
Match: True
```

▲ Figure 9: RoBERTa Model Predictions and Ground Truth Comparison



▲ Figure 10: Training Loss for DistilBERT

In Figure 10, the variation in batch size from 16 to 64 does not seem to have a significant impact on the training loss. All curves, regardless of batch size, follow a nearly identical downward trend, indicating that changing the batch size does not result in noticeable improvements or detriments to the model's ability to minimize loss during training. This suggests that for DistilBERT, batch size is not a critical hyperparameter affecting convergence, at least within the range tested.

In contrast, increasing the number of epochs from 3 to 10 has a clear effect on reducing the training loss. As we increase the number of epochs, the loss consistently decreases across all batch sizes, which is a positive outcome. This indicates that with more epochs, the model continues to learn and generalize better, leading to better performance during training. The continued decline in loss suggests that DistilBERT benefits from additional epochs, reaching lower levels of training loss as the model is exposed to the data for more cycles.

For the results of RoBERTa, we are only showing result of 10 epochs, the results for 3 epochs are present in GitHub repository.

4.1 Conclusion

In summary, ALBERT achieved an accuracy of 43.62% and a macro-average F1-score of 0.43. It excelled particularly in identifying the Reformer label but struggled with the GPT-NEO class due to language complexity. DistilBERT outperformed ALBERT with an accuracy of 67.24% and a macro-average F1-score of 0.67. It showed strong performance on the GPT4o class, indicating its capability in handling nuanced linguistic patterns. RoBERTa trained on 10 epochs demonstrated the highest accuracy of 67.56% and a macro-average F1-score of 0.68. But for 3 epochs its performance was worse than DistilBERT. But this limitation is convered by the fact that RoBERTa with LoRA trained significantly faster than DistilBERT.

5. Analysis / experiments

5.1 Model Performance Comparisons

ALBERT: The ALBERT model, despite its smaller size of 12 million parameters, demonstrated a reasonable balance between accuracy and resource efficiency. It achieved an accuracy of 43.62%, with a macro-average F1-score of 0.43. Notably, ALBERT showed its strongest performance on the Reformer label, achieving a recall of 0.67 and an F1-score of 0.59, indicating that it was more adept at identifying this class. However, ALBERT struggled with classes like GPT-NEO, which had a lower F1-score of 0.29, likely due to a more complex language style or subtle variations between the generated outputs of the models.

Reformer's superior performance in the classification task can be attributed to its innovative architecture. Specifically, Reformer utilizes locality-sensitive hashing (LSH) for self-attention, which helps it efficiently handle long text sequences by focusing on relevant parts of the input. Reformer incorporates reversible layers that reduce memory consumption, allowing it to maintain more contextual information from the input. These architectural innovations make the text patterns generated by Reformer more distinct and easier for the classifier to recognize. This is reflected in its high recall (67%) and precision (53%), which indicate that the classifier can correctly identify a large portion of Reformer completions while also making accurate predictions. The result is a balanced F1-score of 0.59, the highest among the models tested, showing Reformer’s robustness in this classification task.

On the other hand, GPT4o and GPT_NEO performed poorly, each for different reasons. GPT4o, a sophisticated generative model likely based on GPT-4, had a high precision of 0.59 but a very low recall of 0.28. This indicates that while the classifier was often correct when it predicted GPT4o completions, it missed many actual GPT4o instances. The complex, contextually rich completions produced by GPT4o may have blurred the lines between it and other models, making it harder for the classifier to generalize well across all GPT4o completions. GPT_NEO, an open-source version of GPT models, exhibited both low precision (0.27) and recall (0.31). Its poor performance could be due to lower-quality completions or fewer distinctive features in its text generation, which made it difficult for the classifier to distinguish GPT_NEO completions from those of other models.

GPT2 and Gemini had more moderate performance, with both models showing balanced but lower precision and recall scores. GPT2, as an older and widely studied model, likely generates completions that are moderately distinguishable, allowing the classifier to achieve a middle-of-the-road performance. Gemini produced similar results. The classifier was able to recognize their outputs to a certain extent but struggled with precision and recall compared to Reformer. These differences in performance across the models highlight the impact of architectural and training differences, with Reformer's efficiency and distinctiveness giving it a clear edge in classification tasks, while more complex or less optimized models like GPT4o and GPT_NEO proved more challenging to classify accurately.

DistilBERT: As a smaller version of BERT, DistilBERT was faster to train and optimized for resource efficiency with its 66 million parameters. It achieved a higher accuracy of 67.24%, and a macro-average F1-score of 0.67, showcasing significant improvements over ALBERT. DistilBERT exhibited strong performance on classes such as GPT4o, where it achieved a recall of 0.78 and an F1-score of 0.72, suggesting its ability to discern between nuanced linguistic patterns in the generated text. This performance reflects the trade-off between DistilBERT's larger parameter size and its improved generalization on more complex text patterns.

The results highlight significant differences in the classification performance of the five LLMs. Reformer stands out with the best performance, achieving a precision of 0.95, recall of 0.90, and an F1-score of 0.92. This indicates that the DistilBERT classifier had a relatively easy time identifying Reformer's text completions. The high precision suggests that the classifier rarely misclassified outputs from other models as Reformer, while the strong recall shows it successfully captured most of Reformer's completions.

GPT4o performed well with a precision of 0.67, recall of 0.78, and an F1-score of 0.72. While the classifier correctly identified most GPT4o completions (high recall), its lower precision indicates that completions from other models were sometimes misclassified as GPT4o. This suggests that GPT4o's text generation might share some similarities with other models, leading to more misclassifications. Despite this, GPT4o's overall performance was solid, reflecting its ability to generate text that is generally distinct but occasionally overlaps with other models.

Gemini showed moderate performance with a precision of 0.71, recall of 0.55, and an F1-score of 0.62. The classifier was able to predict Gemini completions accurately when it made predictions (high precision), but it missed a significant portion of actual Gemini completions (lower recall). This suggests that Gemini's text generation might not be as consistently distinctive, resulting in the classifier overlooking many completions. Although it performed better than GPT_NEO and GPT2, Gemini still had significant room for improvement in terms of recall.

GPT_NEO showed relatively poor results, with a precision of 0.52, recall of 0.62, and an F1-score of 0.56. The classifier had difficulty distinguishing GPT_NEO completions, leading to frequent misclassifications. This could be due to GPT_NEO's open-source nature, which might generate less refined and more generic completions, making it harder for the classifier to differentiate. The lower precision and recall highlight the challenge of identifying GPT_NEO outputs with confidence.

GPT2 performed the worst, with a precision of 0.56, recall of 0.51, and an F1-score of 0.54. The classifier struggled the most with identifying GPT2 completions generates more generic or less complex outputs compared to the newer models. This made GPT2's completions less distinguishable, resulting in frequent misclassifications. Both precision and recall were low, indicating that the classifier not only missed many GPT2 completions but also frequently misidentified outputs from other models as GPT2.

In summary, Reformer outperformed the other models by a wide margin due to its efficient handling of sequences and distinct text generation, while GPT4o also performed well but showed some overlap with other models. Gemini was moderately accurate but suffered from lower recall, while GPT_NEO and GPT2 performed poorly, with the classifier frequently misclassifying their completions. The architectural differences among the models and the complexity of their outputs were key factors influencing the classification performance.

- **RoBERTa:** RoBERTa, fine-tuned using the Low-Rank Adaptation (LoRA) method, displayed the strongest overall performance, with an accuracy of 67.56% and a macro-average F1-score of 0.68 when trained on 10 epochs and about 62.28 % accuracy and 0.63 F1-score when trained for 3 epochs. The larger parameter size (125 million) coupled with the LoRA fine-tuning allowed RoBERTa to capture deeper contextual understanding of the text. The model was particularly successful with the Reformer label, achieving an average recall of 0.91 and an average F1-score of 0.93 for both 3 and 10 epochs, showcasing RoBERTa's ability to distinguish between more subtle variations in text completion.

The results from the RoBERTa classification highlight the strengths and weaknesses in identifying completions from various LLMs. Reformer's superior performance underscores its architectural advantages in generating distinct text, while GPT4o, Gemini, GPT_NEO, and GPT2 faced challenges related to text similarity and generality. These findings emphasize the importance of model architecture and text generation characteristics in enhancing classification accuracy, providing insights into improving the identification of LLM outputs in future research and applications.

The performance of the three classifiers varies significantly due to differences in their architectures and training methodologies. ALBERT recorded the lowest accuracy at 43.62%, struggling particularly with distinguishing outputs from GPT_NEO and GPT4o. Its architecture, which prioritizes parameter efficiency, may lead to challenges in effectively capturing the distinct text patterns needed for accurate classification. As a result, ALBERT's performance suffered, particularly with models that generate more generic outputs.

In contrast, DistilBERT and RoBERTa achieved higher accuracies in the range of 62%-68%, reflecting their superior classification capabilities. DistilBERT benefits from being a lighter version of BERT, maintaining much of the model's effectiveness while enhancing computational efficiency. This results in improved performance, especially for recognizing distinctive text patterns. RoBERTa, known for its robust training strategies, including the use of more extensive datasets and dynamic masking, outperformed both ALBERT and DistilBERT. These differences highlight how architectural design and training approaches significantly influence each model’s ability to identify completions from various LLMs.

The classification performance of the three models exhibits considerable variation, which can be attributed to differences in their respective architectures and training methodologies. ALBERT achieved the lowest accuracy at 43.62%, demonstrating difficulties in distinguishing outputs from models such as GPT_NEO and GPT4o. Its architecture emphasizes parameter efficiency, which may limit its capacity to capture the distinct text patterns necessary for accurate classification. Consequently, ALBERT's performance is compromised, especially when faced with models that produce more generic outputs. DistilBERT functions as a lighter variant of BERT, effectively retaining much of the original model's performance while enhancing computational efficiency. This improvement facilitates the model's ability to recognize distinctive text patterns more effectively. RoBERTa, noted for its robust training strategies—including the utilization of larger datasets and dynamic masking—demonstrates enhanced performance in comparison to both ALBERT and DistilBERT. These findings underscore the significant impact of architectural design and training approaches on the models' capabilities to accurately identify completions generated by various large language models.

5.2 Truncated text generated model

The Reformer model outperformed all others, likely due to the distinctiveness of its generated text compared to the more similar GPT variants and Gemini, which created confusion for the models.

In summary, the experiments and analyses demonstrated that while ALBERT is efficient and lightweight, its smaller size makes it less suited for complex tasks involving subtle differences in generated text. DistilBERT offers a good balance between performance and efficiency, but RoBERTa, particularly when fine-tuned with LoRA though needing more epochs, stands out as the most effective model for this classification task as it needed comparitvely less time to train even when it was trained till 10 epochs. RoBERTa's larger parameter count, coupled with efficient fine-tuning techniques, allows it to capture more nuanced details in the data, resulting in superior performance. Further improvements could be explored by incorporating additional advanced fine-tuning techniques or using ensembles of models for better generalization.

6. Related work

This project is directly related to several key research areas, such as model attribution and detection, text classification and representation, LLM benchmarking and evaluation, and natural language generation (NLG) evaluation. The objective of the project is to build a classifier capable of identifying which LLM generated a specific text completion. This involves detecting model-specific patterns in text outputs, representing and classifying these outputs, and benchmarking models to understand their performance differences. Additionally, evaluating the quality of the generated text is crucial for refining the classifier's ability to differentiate between various LLMs effectively. These research directions provide the theoretical and practical foundation for developing a reliable attribution system for text completions.

6.1 Model attribution and detection

Model attribution and detection involve determining which LLM generated a given text. Since LLMs like GPT, Gemini, and Reformer can produce distinct completions for the same input, developing a system to accurately attribute these variations to their respective models is essential. Antoun et al. (2024) explore cross-model detection, showing that classifiers trained on one LLM can detect texts produced by other models without retraining, offering a versatile approach to model attribution. Gambini et al. (2023) evaluate attribution and deepfake detection using fine-tuned BERTweet and TriFuseNet, emphasizing the use of stylistic, semantic, and contextual features to differentiate human from machine-generated texts.

Shaib et al. (2024) introduce syntactic templates, showing that LLMs frequently reuse patterns from pre-training data. Their analysis highlights how these templates persist through fine-tuning and serve as a key feature for enhancing attribution systems by capturing stylistic memorization. Li et al. (2024) investigate AI-generated text detection in real-world scenarios, finding that out-of-distribution data presents a challenge. They identify perplexity as a valuable feature for model detection across various domains and models. Munir et al. (2021) focus on attributing synthetic text to its source model, with their fine-tuned XLNet (XLNet-FT) achieving 91% to 98% accuracy. They address the difficulty of distinguishing between fine-tuned variants of the same model and propose further research to improve attribution as LLMs evolve.

6.2 Text classification and representation

Text classification and representation is another important research direction for this project, as it provides the foundational methods for distinguishing between different text outputs. A classifier must learn to represent both the input text (xi) and the appended completion (xj) in a way that captures the distinct features generated by different LLMs. Alsmadi and Gan (2019) discuss challenges in short-text classification, noting that text brevity and noise hinder accuracy. They propose solutions such as genetic algorithms and soft computing to improve performance. Bhattacharjee and Liu (2024) explore ChatGPT's ability to detect AI-generated text, finding it excels in identifying human-written content but struggles with machine-generated text. They suggest leveraging this asymmetry for detection. Peng et al. (2004) introduce the Chain Augmented Naive Bayes (CAN) model, which integrates n-gram language models to improve text classification by capturing local dependencies. CAN outperforms or matches state-of-the-art methods in various languages.

Hartmann et al. (2019) evaluate ten text classification methods across social media datasets, showing that machine learning models like Random Forest (RF) and Naive Bayes (NB) outperform traditional and lexicon-based methods. Nigam et al. (2000) use the Expectation-Maximization (EM) algorithm with a Naive Bayes classifier, incorporating unlabeled data to improve classification. They propose extensions to handle real-world data, reducing errors by up to 30% in various tasks.

6.3 Benchmarking and evaluation of LLMs

Benchmarking and evaluation of LLMs are crucial for establishing standardized metrics and datasets to compare their performance, particularly for tasks such as text completion, coherence, and fluency. Understanding model behavior under similar conditions aids in improving classifier accuracy and highlights how updates (e.g., moving from GPT-3 to GPT-4) can affect attribution accuracy due to changes in generation patterns. Jawahar et al. (2020) review research on detecting machine-generated text, focusing on the increasing challenge as text generation models improve. They categorize detectors into classifiers trained from scratch, zero-shot classifiers, fine-tuning neural language models, and human-machine collaborations. Their error analysis outlines the key difficulties in developing effective detectors for machine-generated content. Valiaiev (2024) discusses the complexities of evaluating text quality from natural language generation (NLG) systems.

The study examines various methods, including comparisons with gold standards, linguistic analyses, expert reviews, and task-based evaluations, while addressing challenges like reviewer inconsistency. Valiaiev suggests using a Turing-style test to provide a more reliable and independently verifiable evaluation method. Varshney et al. (2020) frame machine-generated text detection as a hypothesis-testing problem, analyzing error rates in relation to perplexity. They propose incorporating semantic side information into detection algorithms to improve accuracy and explore how detection algorithms can help attribute generated text to specific models. Lippi et al. (2019) compare statistical features of text generated by LSTMs with human language and Markov models.

They find that LSTM-generated texts exhibit long-range correlations similar to human text and identify an optimal "temperature" parameter for generation quality. The study also touches on authorship attribution, offering insights into how statistical analysis can inform model attribution strategies. Ribeiro et al. (2016) introduce LIME (Local Interpretable Model-agnostic Explanations), a method that explains classifier predictions by creating locally interpretable models. They propose SP-LIME to give a global view of model behavior by selecting representative predictions.

6.4 Natural language generation (NLG) evaluation

NLG evaluation is directly relevant because our project revolves around the task of generating and completing text sequences. Evaluating the quality of text generation from LLMs, such as coherence, relevance, and fluency, will help in fine-tuning the classifier to detect nuanced differences in generated content. NLG evaluation provides insights into the qualitative aspects of model outputs, which can aid in feature selection and the development of more sophisticated attribution techniques. Santhanam and Shaikh (2019) discuss NLG challenges, especially in dialogue systems, emphasizing the need for coherence. They identify future research areas: larger context integration, personality incorporation, and addressing generic responses.

Dong et al. (2022) review advancements in NLG, highlighting data-to-text and text-to-text generation through deep learning, while addressing evaluation challenges and emerging issues from integrating NLG with other fields. Azhar and Nazir (2024) emphasize NLG's role in human communication, discussing trends like neural networks and emotional intelligence. Their insights are vital for effectively evaluating LLM outputs. Khurana et al. (2023) provide an overview of NLP's evolution, applications, and challenges, offering context for the implications of NLP developments on NLG and model attribution. Holtzman et al. (2019) tackle decoding issues in neural language models, introducing Nucleus Sampling to improve text quality by sampling from a dynamic subset of tokens, showing superior results over traditional methods.

References

Antoun, W., Sagot, B., & Seddah, D. (2023). From text to source: Results in detecting large language model-generated content. arXiv preprint arXiv:2309.13322.

Gambini, M., Avvenuti, M., Falchi, F., Tesconi, M., & Fagni, T. (2023). Detecting Generated Text and Attributing Language Model Source with Fine-tuned Models and Semantic Understanding. In IberLEF@ SEPLN.

Shaib, C., Elazar, Y., Li, J. J., & Wallace, B. C. (2024). Detection and measurement of syntactic templates in generated text. arXiv preprint arXiv:2407.00211.

Li, Y., Li, Q., Cui, L., Bi, W., Wang, Z., Wang, L., ... & Zhang, Y. (2024, August). Mage: Machine-generated text detection in the wild. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 36-53).

Munir, S., Batool, B., Shafiq, Z., Srinivasan, P., & Zaffar, F. (2021, April). Through the looking glass: Learning to attribute synthetic text generated by language models. In Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume (pp. 1811-1822).

Alsmadi, I., & Gan, K. H. (2019). Review of short-text classification. International Journal of Web Information Systems, 15(2), 155-182.

Bhattacharjee, A., & Liu, H. (2024). Fighting fire with fire: can ChatGPT detect AI-generated text?. ACM SIGKDD Explorations Newsletter, 25(2), 14-21.

Peng, F., Schuurmans, D., & Wang, S. (2004). Augmenting naive bayes classifiers with statistical language models. Information Retrieval, 7, 317-345.

Hartmann, J., Huppertz, J., Schamp, C., & Heitmann, M. (2019). Comparing automated text classification methods. International Journal of Research in Marketing, 36(1), 20-38.

Nigam, K., McCallum, A. K., Thrun, S., & Mitchell, T. (2000). Text classification from labeled and unlabeled documents using EM. Machine learning, 39, 103-134.

Jawahar, G., Abdul-Mageed, M., & Lakshmanan, L. V. (2020). Automatic detection of machine generated text: A critical survey. arXiv preprint arXiv:2011.01314.

Valiaiev, D. (2024). Detection of Machine-Generated Text: Literature Survey. arXiv preprint arXiv:2402.01642.

Varshney, L. R., Keskar, N. S., & Socher, R. (2020, February). Limits of detecting text generated by large-scale language models. In 2020 Information Theory and Applications Workshop (ITA) (pp. 1-5). IEEE.

Lippi, M., Montemurro, M. A., Degli Esposti, M., & Cristadoro, G. (2019). Natural language statistical features of LSTM-generated texts. IEEE Transactions on Neural Networks and Learning Systems, 30(11), 3326-3337.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016, August). " Why should i trust you?" Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 1135-1144).

Santhanam, S., & Shaikh, S. (2019). A survey of natural language generation techniques with a focus on dialogue systems-past, present and future directions. arXiv preprint arXiv:1906.00500.

Dong, C., Li, Y., Gong, H., Chen, M., Li, J., Shen, Y., & Yang, M. (2022). A survey of natural language generation. ACM Computing Surveys, 55(8), 1-38.

Azhar, U., & Nazir, A. (2024, May). Exploring the Natural Language Generation: Current Trends and Research Challenges. In 2024 International Conference on Engineering & Computing Technologies (ICECT) (pp. 1-6). IEEE.

Khurana, D., Koli, A., Khatter, K., & Singh, S. (2023). Natural language processing: state of the art, current trends and challenges. Multimedia tools and applications, 82(3), 3713-3744.

Holtzman, A., Buys, J., Du, L., Forbes, M., & Choi, Y. (2019). The curious case of neural text degeneration. arXiv preprint arXiv:1904.09751.