Unveiling AI-Generated Text: A Comprehensive Analysis of LLM Outputs and Detection Methodologies

-Soham Barman

This report presents a comprehensive analysis of AI-generated text, focusing on the detection and classification of outputs from various Large Language Models (LLMs). The study employs a custom BERT-based classifier to distinguish between texts generated by GPT, GPT-Neo, OPT, Falcon, and Bloom models. A dataset of 1000 prompts and their corresponding outputs from each model was curated and analyzed for linguistic features, structural patterns, and stylistic elements. The research explores multiple approaches to text classification, including experiments with different input configurations and model architectures. Despite the sophisticated methodology, the classifier's performance remained limited, highlighting the complexity of the task and the need for further research in this area.

Introduction

The large language models have emerged as some of the strongest tools in the ever-evolving landscape of artificial intelligence for generating human-like text across various domains. Whereas such models stand at a remarkably sophisticated level already, determining whether content is AI- or human-generated has become an increasingly important skill among researchers, educators, and content moderators alike. Some of the topics in this report look into the multifaceted aspects of AI-generated text regarding its creation and detection, also considering the implications in academic and research environments.

The work is prefaced with a critical review of the literature, discussing recent efforts related to AI-generated text detection, enhancement of academic reading, and assessment of LLMs for research-oriented tasks. Further to this grounding, we contribute an independent exploration into the nature of text produced by such prominent LLMs as GPT, GPT-Neo, OPT, Falcon, and Bloom.

Our approach is based on the collection of an input dataset, from a variety of LLMs along with their outputs. It is also possible to perform heavy analysis on linguistic features, structural patterns, and stylistic elements that discriminate against different LLM outputs. This forms the basis for a custom classifier, based on the BERT architecture, which is later used to classify the source model of given text samples.

The report also touches on the challenges that were faced in the classification task and gives an overview of possible improvements that could be implemented for further research. This paper contributes to the ongoing debate about AI ethics in content creation and preserving information integrity in the digital era by investigating subtleties in AI-generated text and its methodology of detection.

Discussion of Related Works:

This literature review discusses four recent works, which are going to delve into the diverse aspects of AI in text-related tasks: detection of AI-generated content, assistance in academic reading, and the evaluation of the LLM capability to perform research-oriented tasks. Taken together, the selected studies contribute to our present understanding of the strengths and weaknesses of AI with regard to text processing and generation, and its possible uses within academic and research settings.

1. Detection of AI-Generated Text:

Abburi et al. [1] present a study entitled "Generative AI Text Classification using Ensemble LLM Approaches," the focus of which is centered around finding and attributing AI-generated text with the use of ensemble approaches with Large Language Models. The authors propose a novel approach that couples advanced LLMs with classic machine learning techniques, hence achieving state-of-the-art performance in human vs. AI-generated binary classification and multi-class attribution tasks.

It used the datasets provided by the shared task AuTexTification, including both English and Spanish. This work is multilingual because a number of studies are essentially an investigation of whether such proposed methods generalize to other languages. The key idea in the authors' ensemble neural model is to generate probabilities from multiple pre-trained LLMs and use them as features for a Traditional Machine Learning classifier.

This model performed impressively on the binary classification task, ranking fifth for English texts with a macro F1 score of 0.733 and thirteenth for Spanish texts with a score of 0.649. On the model attribution task, their approach took the first position for both English and Spanish texts, attaining macro F1 scores of 0.625 and 0.653 respectively. Of particular interest is the ensemble approach of Abburi et al. Each input text passes through variants of pre-trained LLMs, fine-tuned on the training data. Then the classification probabilities generated by these models are either concatenated or averaged to create a feature vector that will serve as an input for training TML models for final predictions.

In the same line of thought, Mo et al. [2] introduce the paper "AI text generation detection based on transformer deep learning algorithm," where they introduce a transformer-based deep learning model in the detection of AI-generated text. The authors develop and assess a tool that improves the accuracy of identifying large language models-produced text.

Mo et al.'s approach lies in a comprehensive text pre-processing pipeline and hybrid deep learning architecture. Among these, the preprocessing steps involve unicode normalization, conversion to lowercase, removal of non-alphabetic characters and punctuation, and many other cleaning techniques. This heavy pre-processing methodology aids in normalizing the input text and removing any potential noise or inconsistencies that may impact model performance.

The core of the detection system was a deep learning model that combined several neural network architectures, which will include embedding layer, bidirectional LSTM layer, custom TransformerBlock module, Conv1D, Global max pooling, fully connected Dense layers with dropout and finally another Dense layer, but this time with sigmoid activation for binary classification. This hybrid architecture leverages the strengths of different neural network types-LSTM for capturing sequential dependencies, Transformer for modeling long-range relationships in the text, and CNN for extracting local features.

The model did fairly well on the test set, reaching an accuracy of 99%, precision of 0.99, recall of 1.00, and an F1-score of 0.99. All these metrics definitely substantiate the efficiency of the model in terms of distinguishing between human-written and AI-generated text.

Both research papers significantly contribute to the development of detecting AI-generated text, each from a different approach but with high accuracy. Abburi et al. prove that this ensemble method-the power of combination among several models and techniques-can be top-notch, whereas Mo et al.'s hybrid architecture proves to be highly effective by including different neural network

types. These approaches are important tools in dealing with the increasing sophistication of language models and contribute to information security and content authenticity in a number of domains.

2. Augmented Reading of Academic Text:

Richards [3] introduce the enhanced reading interface ReaderQuizzer, which supports improving reading comprehension and engagement with academic texts. This tool automatically generates learning questions that foster better understanding and develop the skills of critical reading by encouraging active engagement with the text.

With respect to their analysis, the authors point out several key issues that lie at the root of the challenges related to academic reading. According to them, these are:

1. Inability to read at a collegiate level

2. Time constraints and institutional lack of support

3. Poor compliance with required readings

ReaderQuizzer seeks to overcome these barriers by embedding automatically generated learning questions within academic documents. The research methodology for this study included literature review, early prototype development, pilot study, redesign and implementation and final evaluation study.

Six principles influenced the design of ReaderQuizzer:

1. Provide an ideal quantity of questions

2. Provide question types

3. Linking users to answers of generated questions

4. Relevance and reduced repetition of questions

5. Variations in placing questions as per the structure of an article

6. Note-taking workflow

The study of the evaluation conducted on ReaderQuizzer was done on a group of 16 undergraduate students from a few disciplines. The overall outcome was good; however, participants reported that their reading comprehension skills were improved, and the quality of generated questions was good, thereby motivating people to complete the readings as shown in the outcome.

3. Evaluating LLMs for Research Tasks:

In the work of Kang and Xiong [4], a benchmark called ResearchArena is presented for assessing the performance of LLMs on academic survey tasks. This is very important because it might revolutionize how research is currently done, most especially in fast-growing areas where information overload inhibits understanding.

ResearchArena deconstructs the academic survey process into three main stages:

1. Information Discovery: Locating relevant papers

2. Selection: The rating of the importance of papers to the topic at hand

3. Organization: The organization of the papers into meaningful structures

The benchmark is created based on a large dataset comprising 12.0 million full-text academic papers, 7,952 survey papers, and 1,884 mind-maps extracted from the surveys. It is elaborately curated from the Semantic Scholar Open Research Corpus (S2ORC), which ensures its high reliability and scholarly relevance. The authors provide preliminary evaluations based on several baseline methods, including state-of-the-art naive keyword-based and more recent LLM-based methods. Most notably, the following findings have emerged:

Information Discovery remains hard for all baselines; their Recall@100 was less than 0.15 and 0.27 for BM25 and BGE retrievers.

Information Selection: keyword-based methods tend to outperform the agent baselines steadily.

Information Organization: the metrics are inconsistent under different sets of ground-truth criteria; CLUSTERING generates the best hierarchies, with the LLM-based agents outperformed.

The papers discussed within the framework of the literature survey represent landmark achievements in AI applications concerning text-related operations. While Abburi et al. and Mo et al. demonstrated the effectiveness of ensembles and hybrid techniques, respectively, in detecting AI-generated text-effectively arming ourselves in the interest of content veracity, Kang and Xiong showed the use of AI-aided tools in improving academic reading experience by tackling some long-standing higher education challenges in their work on ReaderQuizzer. The benchmark by Kang and Xiong, called ResearchArena, finally laid down a comprehensive assessment framework on whether LLMs are up to the task of carrying out academic research work; hence, this work brings out both the promise and limitations of these models.

Put together, these studies add to the emerging understanding of the current capabilities and limits of AI with respect to text processing and generation, and their possible applications in academic and research contexts.

Dataset Curation

LLMs Used for Text Generation

The following models were choses for the purpose of text generation: GPT-Neo, EleutherAI/gpt-neo-2.7B, Falcon-7B, tiiuae/falcon-7b, BLOOM, bigscience/bloom-560m, GPT-2, gpt2, and OPT, facebook/opt-1.3b, great care was taken to provide a good model performance balanced with computational feasibility. Considering large-scale models cannot be used due to the constraints my system has, the choice fell on lightweight models, chosen strategically to show better results in text generation without claiming too much computational resource.

The GPT-Neo model, EleutherAI/gpt-neo-2.7B, is an open-source alternative developed by EleutherAI to the large proprietary models such as OpenAI's GPT-3. Compared to other family members of GPT, this 2.7 billion parameters version of GPT-Neo is relatively lightweight, which provides a good trade-off between the efficacy of generated text and hardware requirements. The model presents pretty good performance in text completion, being robust enough for most language modeling applications, and should be considered when system memory and processing power are at a premium.

The Falcon-7b is one of the most recent lightweight model architectures with 7 billion parameters. This model has set a bar new in top performance for the smaller LLMs; it yields competitive results without the interference of huge infrastructure. The architecture is optimized for efficient inference so that text will be generated really fast. Although relatively small in size, Falcon-7B has shown

excellent performance on a variety of NLP benchmarks and therefore should generate coherent, contextually relevant text.

BLOOM-560M, as you might guess from its name, is a smaller variant of the BLOOM model, and that makes it particularly useful to those with less computational power. While it has just 560 million parameters, it's still relatively light and does a good job with text generation. Another great application of BLOOM would be its multilingual support-there lies the importance of this model for generating text in several other languages and contexts. Its compactness ensures times of fast inferences and low usage of memory, making it ideal in those specific scenarios where computational efficiency remains key.

GPT-2 developed by OpenAI has long been the centerpiece for Natural Language Generation tasks. Having 1.5 billion parameters, GPT-2 strikes a perfect balance between quality and computational demands. Although improved models have come out, GPT-2 has remained very relevant simply for having such a broad scope of capabilities with relatively low latency. This, together with the efficiency of its scaling, makes the model very desirable in highly constrained environments, since versatility and familiarity with GPT-2 are assured.

Finally, the model OPT-1.3B, developed by Meta, stands out as an especially promising alternative to be much lighter regarding the execution of code. With competitive performance in text generation and having 1.3 billion parameters, this model was designed to be light regarding computational tasks, avoiding heavy power from GPUs or CPUs. OPT has been optimized for speed and inference efficiency, making it ideal for generating text when the system resources are not unlimited.

These five models were selected to balance the trade-off among model size, computational efficiency, and generation capability. They realize viable solutions of high-quality text generation without requiring extensive computational resources from an enormously larger model. Therefore, they are the best suit in an environment with system constraints.

Input Prompts(xi) and Output Result(xj) Generation

To create representative and diverse input data, xi, a hybrid approach was used in the preparation. First, 50 sentences were handcrafted to capture a wide range of contexts and incomplete phrases. These sentences, like "Yesterday I went," "The cat jumped,," "She opened the,", "In the morning," and "He decided to," were all truncated to elicit further text generation. To expand this set and provide the models with a larger input space, an additional 950 truncated texts were created using Claude 3.5, which is a large language model. These were similarly structured to the handcrafted sentences, expanding the set to 1000 prompts in total. Asking Claude 3.5 to create additional inputs supports the great variety of text structures and linguistic nuances in the dataset, helping to test the generalization and performance of the text generation LLMs.

In retrospect, while crafting the input sentences (xi) for the text generation models, another viable approach would have been to scrape grammar books for example sentences. This method could have provided a wide array of structured, contextually appropriate sentence fragments that align with established grammatical rules. By extracting sentences from such sources, the dataset could be enriched with inputs reflecting formal language use across different contexts, without relying on another large language model (LLM) like Claude 3.5.

For the creation of xj across different models, I used the pipeline provided by Hugging Face for generating text. Each model was initialized in a pipeline intended for text generation; thus, the task was specified as producing continuations of input text, xi. Since the pipelines provided a common

framework that would allow efficient text generation among models with consistency, each model was initialized in a pipeline for text generation.

Every prompt (xi) passed to the five initialized pipelines of the models is going to create a text completion, xj,. This performs text generation using the model pipeline. It sets the maximum value to 50 tokens for generated text so consistency is maintained in output for each model. Any glitches that are sent back from this level, even those modelled poorly, are captured and treated here without the process being entirely hindered.

I added a checkpointing mechanism that will allow interruptions and resumptions without loss of information. The checkpoint file keeps track of pairs of input texts, xi, dealt with and the corresponding LLM models. Since computational efficiency may be an issue here and there could be unforeseen errors along the way in processing, I added exception handling to allow the generation of text from the other models when some models may fail.

Analysis and Breaking down into different datasets for experimentation

In this section, I conduct a comprehensive analysis of the outputs generated by GPT, GPT-Neo, OPT, Falcon, and Bloom which vary in terms of architecture, size, and training data, to understand the significant differences in the outputs they produce to understand the distinctive linguistic features and patterns that can be leveraged for classification tasks.

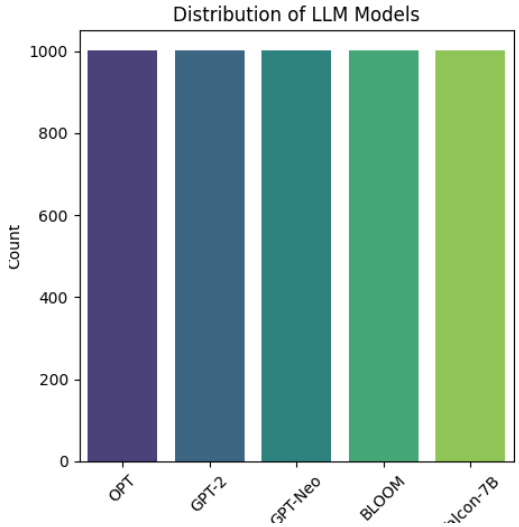
The dataset created in the above section comprises responses generated by LLMs such as GPT, GPT-Neo, OPT, Falcon, and Bloom. Each model was provided with a common set of input prompts (xi), and the resulting outputs (xj) were collected for analysis. A key goal in constructing the dataset was to maintain balance, ensuring that no single model dominated the dataset in terms of response volume.

The primary fields in the dataset include:

- **xi**: The input prompt provided to the LLM.
- **xj**: The generated response by the LLM.
- **LLM_model**: The label indicating the model responsible for generating the response.

Distribution of Responses Across LLM Models

First, a bar chart was created to show the number of different responses for each of the LLM models, showing quite well that the dataset is balanced-between the numbers of responses coming from different models. A balanced dataset is very important in ensuring that any future machine learning classifier is not biased toward the outputs of one of these models. The balance allows more robust results, since the classifier will have equal chances of learning from all involved models.

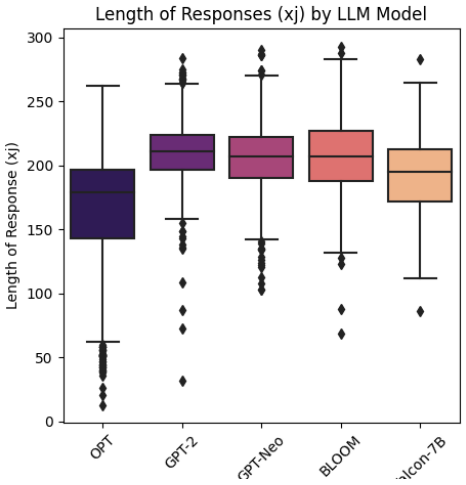


Response Length by LLM Model

Perhaps one of the most straightforward but also enlightening metrics for understanding model behavior is that of response length. A comparison boxplot was created, showing response length (in characters) across various models.

- GPT: Shown to produce longer responses, usually more elaborate and with more details.
- GPT-Neo: Produced outputs slightly shorter than GPT, but still relatively verbose, indicating a preference for more detailed responses.
- OPT: This model produced shorter responses and thus more concise than GPT.
- Falcon: Its response length was moderate, right between GPT and OPT.
- Bloom: Varied in response length but generally tended to be concise.

This may reflect differences in the model architecture and their training objectives. For example, GPT models are trained to generate outputs that are more detailed and fine-grained, hence their wordier outputs, while models like OPT are optimized for efficiency and generate far shorter responses.

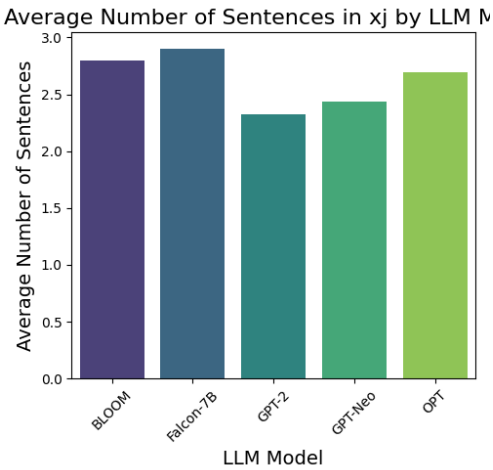


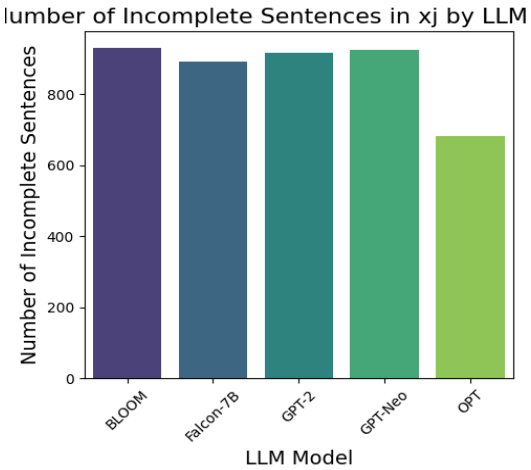
Sentence Structure and Complexity

In addition to length, the structure of the responses was examined by calculating the number of sentences in each response. A custom function was used to detect punctuation markers (periods, exclamation marks, question marks) to estimate sentence count.

- GPT and Falcon: Frequently generated multi-sentence responses, reflecting a more complex and structured approach to answering prompts.
- GPT-Neo: Displayed similar behavior to GPT, frequently generating multi-sentence responses.
- OPT: Tended to generate shorter responses with fewer sentences, focusing on brevity.
- Bloom: Showed a mix of short and long responses, with a wider range in the number of sentences per response.

Models that produce multiple sentences, like GPT and Falcon, may be better suited for tasks requiring detailed and elaborate responses. In contrast, simpler sentence structures from OPT reflect a design focused on generating more direct answers. Sentence count, therefore, serves as another valuable feature for distinguishing between LLM outputs.





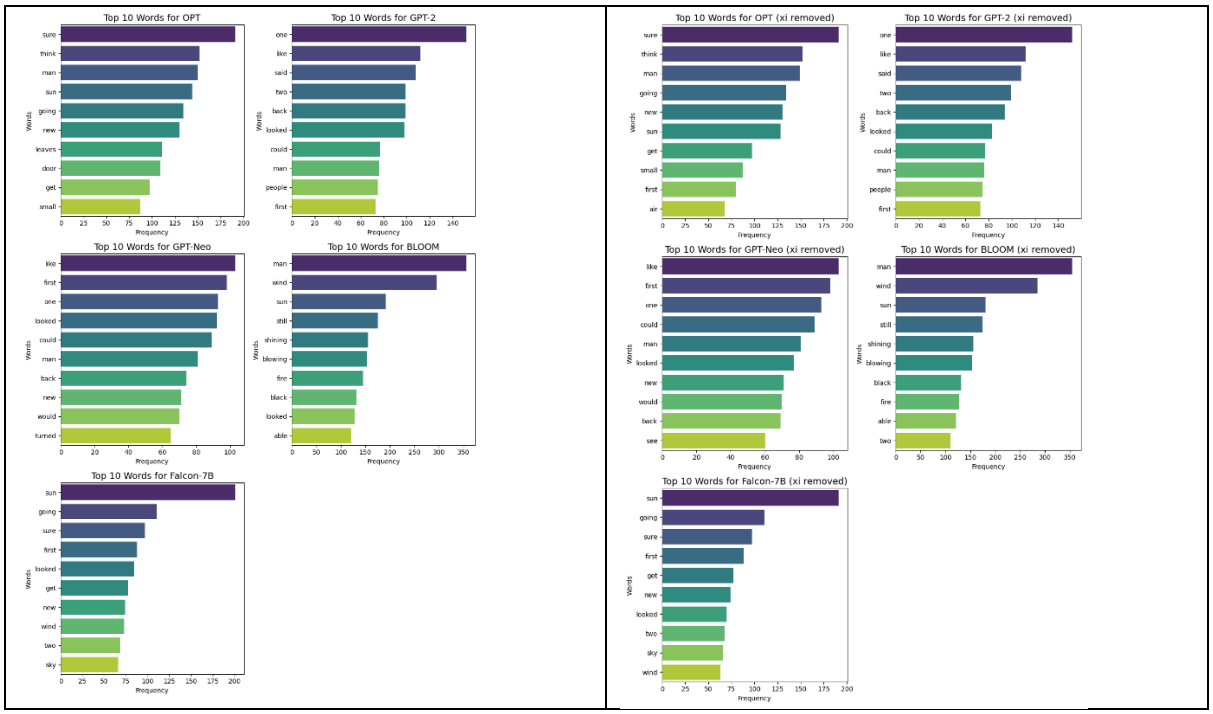
Top 10 Words Produced by Each Model

A lexical analysis to identify the top 10 most frequently occurring words for each model. This analysis was performed both when the input prompt (xi) was present in the response (xj) and when it was not.

- If xi is present in xj: Models like GPT and GPT-Neo frequently included portions of the input prompt in their outputs, which was reflected in the top 10 words. These models often adhered closely to the input, generating coherent responses that aligned with the prompt content.
- If xi is not present in xj: In the absence of the input prompt, the top 10 words revealed the models' inherent tendencies. GPT-Neo, like GPT, produced more abstract, thematic words, while Falcon and OPT showed a tendency toward factual and concise word choices.

Whether or not the input prompt is reflected in the response can influence a model's lexical choices. Models like GPT and GPT-Neo tend to generate more verbose, abstract responses when the input is absent, whereas more factual models like Falcon stick to concise language. These differences in word

usage are valuable features for classification tasks. This Division Formed a basis of dividing our data further for expermentation



Unique Prompt ID Generation

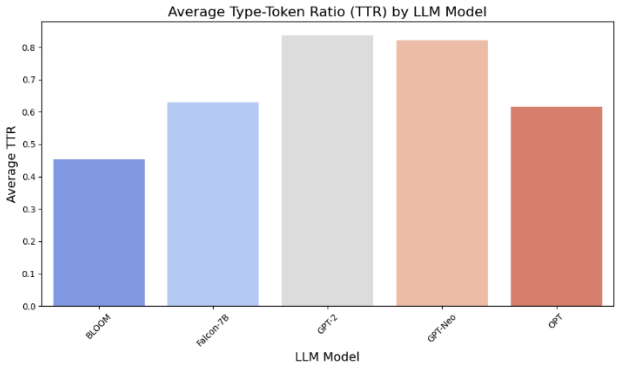
For the tracking of responses across models uniformly, each of the input prompts has been allocated a unique ID number, or prompt_id. The system of unique prompt IDs allows for the comparison of model outputs in a more strenuous and precise manner. The consistency in which we track each input prompt across models permits us to directly compare how each LLM treats the same input and uncover tendencies specific to the individual model.

Average Type-Token Ratio (TTR) by LLM Model

TTR calculates the lexical diversity by using the number of unique words or types and the total amount of words or tokens. The richness of vocabulary for each model was determined by this metric. The results are as follows:

- GPT: Returned high TTR values from the chosen texts, which were highly diverse in vocabulary.
- GPT-Neo: Returned a TTR value not far from that of GPT. Hence, it also supported the Nuanced use of Language.
- OPT: TTR value was low, indicating this model had repeated certain words more.
- Falcon: Had an average TTR, neither too repetitive nor too creative.
- Bloom: Had more lexical variation but tended to be on the more repetitive side.

Models such as GPT and GPT-Neo have been seen to give lexically richer outputs due to their higher values of TTR. Models such as OPT, however, are more focused on efficiency and resort to lesser usage of vocabulary. This may make TTR a helpful feature in setting these models apart based on the richness of the generated language.

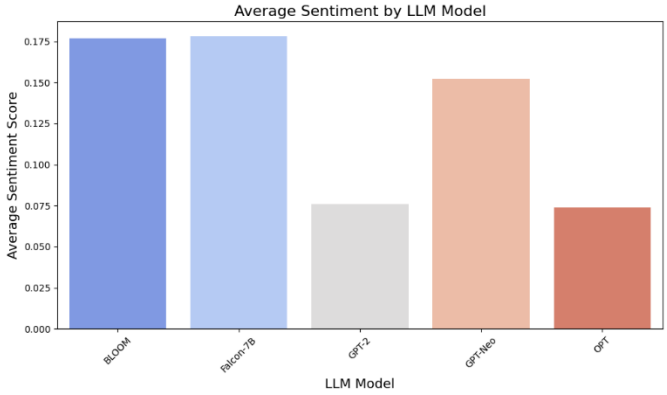


Sentiment Analysis by LLM Model

The sentiment analysis carried out on the outputs established what kind of emotional tone was depicted for every response. The scores of sentiments range from negative, showing negative sentiment, to positive, showing positive sentiment.

- GPT : Generally came out as neutral or slightly positive in sentiment in their responses.
- OPT: Much more neutral sentiment, with less deviation between the outputs.
- Falcon: Its responses were slightly positive in sentiment, very close indeed to those of Bloom.
- Bloom: Displayed more varied sentiment, with outputs ranging from positive to negative.

The interpretation henceforth would be that SA depicts the emotional quotient of models when input prompts are provided. GPT and OPT, having somewhat positive sentiments, may be put to work on tasks which require neutral or positive emotional tone. Models like that of Bloom, which express varied sentiments, might perform better on tasks that require nuance in emotions.



N-gram Analysis: Frequently Repeated Phrases

N-gram analysis-unigrams, bigrams, and trigrams-was done to identify phrase repetition in the generated outputs. In this context, it provides a better idea of which linguistic structure is preferred by each model.

- GPT and GPT-Neo: High frequency phrases generated, like multi-word phrases such as "the result is" or "this suggests," reflecting that their language is structured and academic-like.
- OPT: It constructed simpler, more repetitive phrases, focused on communicating directly and briefly.
- Falcon: It showed both short and informative phrases.

- Bloom: Varied n-grams, with partial congruence to GPT Neo on academic phrase usage, though more conversational.

The analysis done on GPT and GPT Neo, show that these models are really set on making structured and formal phrases, as was meant for this sophisticated speech, while OPT produces simpler and more straightforwardly phrased sentences.

OPT	**GPT-2**	**GPT-Neo**	**BLOOM**	**Falcon-7B**
(sun shining, 57)	(looked like, 13)	(crowd cheered, 14)	(wind blowing, 188)	(sun shone, 47)
(best best, 44)	(moon rose, 9)	(new york, 13)	(sun shining, 162)	(wind blew, 41)
(leaves grew, 44)	(don know, 8)	(cheered crowd, 12)	(wearing black, 79)	(sun setting, 36)
(crackled crackled,40),	(door opened, 8)	(united states, 12)	(blowing direction, 69)	(sun shining, 36)
(grew leaves, 40)	(sun set, 8)	(looked like, 9)	(stars shining, 52)	(didn mean, 27)

This table summarizes the most frequent 2-grams generated by each LLM model, highlighting certain trends, such as natural descriptions like "sun shining" being common across multiple models, and domain-specific phrases such as "new york" and "united states" appearing in GPT-Neo's output.

Datasets Created From this Analysis

Beyond simple length and sentence structure, this dataset underwent further transformations in preparing it for subsequent experimental work. The particular transformations involved:

a) Exclusion of cases where the generated response xj may have included part of the input prompt xi.

b) Exclusion of other responses beyond the first sentence so as to retain the immediate responses that each LLM generated.

First, the filtering of responses to include only fully formed, grammatically complete sentences-this transformation eliminates incomplete or truncated outputs, which may happen with some models at generation time. Filtering based on parts of speech, in order to isolate certain grammatical structures and analyze their presence across various models.

Each of these transformations shows different features of the generated responses and will form the basis of experiments where the classifiers are trained to discriminate between LLMs on the basis of stylistic and structural features.

Classification Model

Model Overview

I implemented a custom classifier based on the BERT (Bidirectional Encoder Representations from Transformers) architecture, specifically utilizing the 'bert-base-uncased' pre-trained model as our foundation. Our model, EnhancedModel, is an extension of the basic BERT architecture to add more layers to improve the performance for our specific classification problem. The EnhancedModel therefore had the pre-trained BERT model with a dropout layer on top for regularization and avoidance of overfitting at p = 0.2. This is followed by two fully connected layers with 512 and 256

units, respectively, both using ReLU activation functions. The final layer is a classification head that outputs probabilities for each class in our multi-class problem. This architecture was chosen to balance the transfer of pre-trained knowledge from BERT with task-specific learning through the additional layers.

The mini-batch training was done with a batch size of 64, iterating for up to 20 epochs. An AdamW optimizer, a weight decay regularization correctly implemented variant of Adam, was used during training. The initial learning rate was set to 3e-5, and the weight decay to 1e-4, typical values when fine-tuning transformer models on classification tasks. As part of our effort to further optimize training dynamics, a linear learning rate scheduler was implemented with a warm-up in the first 10% of the training steps; this is a known technique to improve the convergence of transformer models. The AdamW Optimizer variant rectifies an important shortcoming of the original Adam optimizer, which had to do with the interaction between weight decay and adaptive learning rates. AdamW can regularize more effectively by decoupling the weight decay from the adaptive learning rate mechanism. This would be particularly important in the fine-tuning of large pre-trained models like BERT, as the careful balance between keeping the learned representations and adapting to the new task will be tricky.

A number of strategies were employed to enhance generalization and prevent overfitting: I set early stopping with patience of 3 epochs, monitoring validation loss as the criterion for stopping. This will prevent overfitting and ensure that once the performance of the model on the validation set starts to get worse, the training will stop. Besides that, I used a gradual unfreezing approach: freezing the BERT layers while adapting the task-specific layers and then fine-tuning the whole model. This helps in improving performance on transfer learning with big pre-trained models by allowing the new layers to learn meaningful representations before fine-tuning the whole model. This is further augmented by the scheduler, providing a very robust optimization strategy together with AdamW. The learning rate during warm-up will increase from a very small value up to our set initial learning rate of 3e-5. This allows the model to gradually adapt to the new task without extreme changes to the pre-trained weights. After the warm-up, the learning rate linearly decays, which helps in fine-tuning the model as it converges to an optimal solution.

Model evaluation was performed based on some metrics: loss, accuracy, and Area Under the Receiver Operating Characteristic curve. Such a metric has been computed on a held-out validation set after every epoch to guide both the early stopping mechanism and model selection. The use of AUC-ROC, in particular, will provide a threshold-independent measure of the model's discriminative power, something that is going to be especially valuable in multi-class classification scenarios, where class imbalance can easily occur.

The performance of the final model was evaluated on a separate test set to provide an unbiased estimate of the model's generalization capability. Although not used for the final training, I implemented a thorough hyperparameter tuning function able to perform a grid search over learning rates, batch sizes, weight decay values, and warm-up steps. This hyperparameter tuning function allows us to make a systematic study of the parameter space that may reveal better combinations for some datasets or classification tasks.

Our methodology represents how well the transfer learning from large-scale pre-trained models can be used for any specific text classification task. In this work, I investigate the usage of BERT embeddings as an input to the task-specific layers and adopt various advanced training techniques to achieve state-of-the-art performance in multi-class text classification while easing some of the common pain points like overfitting and optimization issues with deep neural networks. Using BERT

as the base for our proposed model leverages unsupervised pre-training on large corpora of text to learn deep semantic representations of input words. Pre-training allows the model to learn complex patterns of language and semantic relationships that can later be fine-tuned for particular downstream tasks, such as our multi-class classification problem.

Results

Dataset Configuration	Test Loss	Test Accuracy	Test AUC-ROC
Original Dataset	1.6799	0.1557	0.5120
Removed xi from xj	1.6861	0.1577	0.5128
xj contained only completed sentences	1.6872	0.2020	0.4833
xj limited to a singular sentence	1.6027	0.2156	0.5525
xj contained xi and was limited to completed sentences	1.6776	0.1377	0.4966
xj without xi and limited to the first sentence	1.6065	0.2056	0.5352
xj composed of parts of speech (Noun and Verb)	1.6469	0.1836	0.4951

Results Analysis

These results, from these various sets of the classifier model show one thing-that indeed, there is a consistent pattern in the performance that is very limited across the test datasets that the model has not learned effectively from the data and the results are randomized. The overall test accuracy remains low, and the highest was at 0.2156 when xj is limited to a singular sentence. It would then signal a trend whereby the model, for perhaps a number of reasons, is struggling to effectively distinguish between classes.

It could, in fact, be that the task at hand is quite complex. If the model architecture is a bit more sophisticated, encapsulating the inter-relationships between different features involved in generating text pairs will provide better results. This model itself, though capable, may not have the required depth to understand semantic nuances of the kind called for by distinguishing text generated by different LLMs.

What is more, experiments showed that ablation studies removing the prompt part, xi, from xj had no significant gain in either test accuracy or AUC-ROC. The intuition behind this observation was that xi provided contextual clues necessary for the model to learn prompt-completion associations. If xj contained only completed sentences, then the model performances degraded significantly for those settings. As there is no prompt context, it does not allow the model to make much sense out of the given input.

However, it may be also biased to the exact composition of the training data and lose some generalization capability to unseen examples. Besides, the relatively small size of the dataset and the limited variety of queries might prevent the model from capturing the full underlying semantics. A

much bigger dataset with various prompts and completions could give richer information for the model to learn from.

In order to further enhance this, there might be a need to try generating completed sentences that include prompt contexts, which may further put the model in a better position to understand how these prompts affect the generated texts amongst different LLMs. Secondly, making use of more feature-rich model architectures- probably fine-tuned on a larger dataset-would enhance feature extraction and classification accuracy.

Conclusion

The investigation into the classification of AI-generated text reveals that there are lots of thorny issues in telling one LLM output from another. That study engaged very advanced methods, from a special BERT-based classifier to linguistic analysis in detail, but still, results show the complex nature of telling LLM outputs effectively. The output of the classifier, which had an accuracy of only 21.56% when its best performance was realized by limiting inputs to single sentences, would seem to indicate that current methods are not yet cognizant of capturing such fine differences produced by LLMs.

These results really emphasize the need for more sophisticated approaches in the area of AI text detection and classification. Further work might be done on more feature-rich model architectures, further expansion of the dataset with more types of prompts and completions, and possibly further research into alternative ways to capture the semantic subtleties that make one LLM distinct from others in creating outputs. Additionally, the study points out the importance of considering prompt context within a classification task and how this information may be used in order to enhance future models.

References

[1] Abburi, H., Suesserman, M., Pudota, N., Veeramani, B., Bowen, E., & Bhattacharya, S. (2023). Generative AI Text Classification using Ensemble LLM Approaches. ArXiv. /abs/2309.07755

[2] Mo, Y., Qin, H., Dong, Y., Zhu, Z., & Li, Z. (2024). Large Language Model (LLM) AI text generation detection based on transformer deep learning algorithm. ArXiv. https://arxiv.org/abs/2405.06652

[3 Maldonado, L. R. (2023). ReaderQuizzer: Augmenting Research Papers with Just-In-Time Learning Questions to Facilitate Deeper Understanding. ArXiv. https://arxiv.org/abs/2308.07988

[4] Kang, H., & Xiong, C. (2024). ResearchArena: Benchmarking LLMs' Ability to Collect and Organize Information as Research Agents. ArXiv. https://arxiv.org/abs/2406.10291