A Deep Learning Classifier for Identifying Large Language Models Through Text Completion Analysis

Sivani Devarapalli and Dhanush Gurram

Department of Computer Science, Pennsylvania State University

Abstract

This work investigates a deep learning-based approach to distinguishing between different LLMs using their completion regarding a given set of truncated texts, xi. Several LLMs in conjunction are exploited to complete the texts, denoted by xj. We then aim at identifying which LLM has generated each paired xi and xj text completion. We develop a classifier based on transformer-based embeddings that reaches an accuracy of 61.03%. These results point out that identifying an LLM from its completion pattern is possible, based on subtle but identifiable differences within the generation of content generated by different LLMs.

1 Introduction

Large Language Models have indeed revolutionized natural language processing. They are impressively capable of generating text, comprehending it, and even formulating responses. There have been models trained on vast corpora, such as GPT-2, GPT-Neo, Falcon, and Facebook-OPT, LLaMA, Vicuna which are capable of coherent, contextually relevant text completions whenever partial prompts are available. While these remain widely used, the ability to attribute distinction among such models from their respective generated outputs has grown increasingly relevant. This is important for several reasons: first, it will allow the identification of model-specific biases; second, it will assess the quality of the content; lastly, it will ensure accountability attached to the generated content.

Since their inception, LLM identification from text completions has always been challenging since they have similar training data and architectures, and finally the way they generate languages. Given the same truncated text prompt xi, different LLMs will generally produce different completions xj. These differences are typically fairly subtle and nuanced. This immediately raises a very interesting question: Is it possible to train a model in machine learning that predicts for a given text completion, what is the originating LLM with good accuracy? This is important, as far as NLP is concerned because an adequate explanation of this question insinuates the marked differences that characterize LLMs and enables users and developers to make sense of outputs coming from different models.

In this work, we present the challenge of identification of LLMs using a deep learning-based classification approach. Given a set of truncated texts, each like xi = "Yesterday I went" for example, we adapt

different LLMs to generate completions, xj such as "Yesterday I went to Costco and purchased a floor cleaner." Our goal is to devise a classifier to correctly identify what LLM produced each complete text pair, xi, xj. These are addressed by using transformer-based embeddings of the pairs of text, training a deep learning classifier to learn distinctive patterns and styles specific to different LLMs.

The contributions of this paper are as follows:
1. We present a novel approach for identifying LLMs based on text completion patterns, using a deep learning classifier trained on transformer-based embeddings.
2. We generate a diverse dataset of (xi, xj) pairs using multiple state-of-the-art LLMs, and use this dataset to train and evaluate our classifier.
3. We demonstrate the feasibility of distinguishing between different LLMs, achieving an accuracy of 61.03% on our test set, and provide insights into the characteristics of each LLM that make them identifiable.

The rest of the paper is organized as: Section 2 describes the related works on both LLM identification and text classification. Section 3 describes how the dataset was generated while Section 4 discusses in detail the methodology used along with the feature representation and model architecture. Section 5 describes the experimental setup while Section 6 presents the evaluation metrics. Section 6 presents results and analyses performances from the proposed classifier. Finally, Section 7 concludes the paper by providing some future research directions.

2 Discussion

Particular attention has been given to the identification of generative models, especially Large Language Models, because there is a great need for content attribution and accountability. Most of the previous work focused on distinguishing between AI-generated and human-generated text rather than distinguishing between different LLMs. Most current methods in this direction depend on statistical features, which are not very capable of handling the complexity of modern LLMs.

While deep learning has been especially successful with BERT and other models for text classification, authorship attribution, and stylistic analysis, specific LLMs can be fed the same prompt and retain some subtlety in the text that might help in distinguishing between them. Our work extends the use of deep learning in classifying particular LLMs beyond the common human versus AI classification. To the best of our knowledge, this represents one of the first systematic comparisons of LLMs on grounds of their text completion pattern.

3 Dataset Curation

3.1 Data Generation

We gathered a collection of incomplete texts (xi) and had a few LLMs provide completions (xj). The incomplete text (xi) consisted of prompts like "Yesterday I went," aimed at eliciting different responses from each LLM. We used four advanced LLMs—GPT-2, GPT-Neo, Falcon, and Facebook-OPT—to generate continuations for each prompt, creating a diverse set of completed texts (xi, xj) from each LLM for the same starting prompt.

3.2 LLMs used for sentence completion

The LLMs chosen for this dataset represent a variety of architectures and training corpora, allowing for an analysis of their distinctive characteristics in text generation:

1. GPT-2: Developed by OpenAI, GPT-2 is a transformer-based model trained on a diverse dataset with the ability to generate fluent and contextually coherent text.
2. GPT-Neo: An open-source alternative to GPT-3, GPT-Neo was trained on the Pile dataset, which includes a broad range of text types.
3. Falcon: A highly optimized transformer model that emphasizes efficiency in text generation while maintaining high performance.
4. Facebook-OPT: A model developed by Meta AI that aims to replicate and improve upon the capabilities of OpenAI's GPT-3.

3.3 Data Statistics

We took a curated dataset with 8756 unique prompts xi from sentences from the Harry Potter novel book, each completed by all four LLMs. This would result in a total of 35028 completed text pairs xi, xj. Next, each completed text was labeled depending on which LLM generated the text; this would amount to four classes since there were four different LLMs. This dataset was further divided into training and testing subsets—90% used for training and 10% for testing—while making sure that each subset retained equal representation from all the four LLMs.

3.4 Preprocessing

We have done a number of preprocessing steps for the standardization of the dataset:
1. Normalization: All the text completions were lowercase, and punctuation was standardized to remove formatting inconsistencies.
2. Tokenization: The texts were tokenized with the BERT tokenizer; this preprocessed the text and generated token IDs suitable for this type of deep learning model.
3. Truncation and Padding: Each pair was truncated to fit within the maximum sequence length of 40 tokens and, where needed, were padded to make all sequences the same length.

3.5 Data Labelling

Each completed text is tagged with the identity xi, xj of the LLM from which it was generated. This permits the model to be trained in a supervised way, with the aim of learning to classify each completed text by its underlying LLM. Therefore, there are four classes in total within the dataset—one per generating LLMs.

3.6 Challenges faced in Data curation

The important challenge while curating the datasets was to have enough diversity in the prompts xi to get enough variation in the responses xj from each LLM. The diversity will help the model get enough strength to differentiate between outputs from different models. Some models such as LLaMA-3B, Vicuna-7B, T-5, and Mistral-7B were employed toward generating pairs of (xi, xj). In the limited GPU resources we presently work on, running some of these models proved difficult. Of course, models like LLaMA-3B and T-5 sometimes generate repetitive or irrelevant outputs. Because of this, we have designed the sampling of prompts in such a way that they would vary from narrative and informational to conversational topics, for catching a lot more different genres of text and improving the quality of outputs.

4 Classifier & Training

4.1 Classifier architecture

The classifier architecture works based on LLM text completions observed progressively elicited from input embeddings X = (xi, xj), where xi and xj are

pairs of truncated texts and their respective completions. The embedded input for the classifier will be an input layer embedding extracted from a pre-trained BERT model. This model embeds the text pairs in dense vector representations to capture much semantic meaning, then applies mean pooling to generate a single fixed-size embedding for each pair.

The classifier consists of two fully connected layers after the embedding layer. The first layer is applied to the mean pooled embeddings and is a nonlinear activation function such as ReLU for injecting some nonlinearity into the model.

After the first fully connected layer, add a dropout layer to prevent overfitting by stochastically shutting off a fraction of the input neurons during training. Now, this output is projected into another fully connected layer, further refining this feature representation into dimensions equal to the number of classes of the output. The final part of this architecture is the output layer, which is a softmax output layer that transforms the logits from the last fully connected layer into class probabilities. It does this to make the predictions of the model and provide the probability of which LLM generated the given text completion. This architecture combines the rich contextual embeddings from BERT with a row of dense layers—learning discriminative features on one hand, generalizing well to unseen data, and distinguishing similar outputs generated by various LLMs on the other.

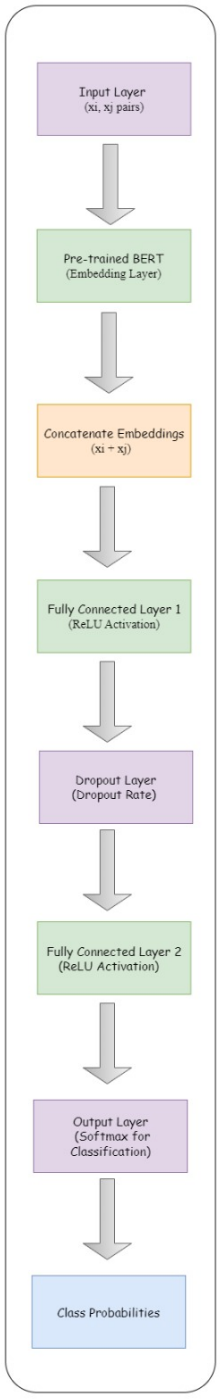

Figure 1: Model architecture

4.2 Training data

The Cross-Entropy Loss function is best fitted for the training process of a classifier in multi-class classification, where it measures the dissimilarity between the predicted probability distribution of class labels and the true labels. This gives a precise measure of how good the correspondences between the predicted

probabilities and the actual correct label are; this hence allows effective learning on identifying which LLM generated a given text completion.

Relevantly, we adopt the AdamW optimizer, which is an extension of the Adam optimizer with weight decay added as a regularization method. AdamW adapts the learning rate for each parameter based on the first and second moments of the gradients. The learning rate is set to 5e-4, which has been empirically validated for fine-tuning transformer-based models like BERT. Besides this, we utilize a linear scheduler with a warmup—a strategy of increasing the learning rate during the initial training and then linearly decaying it to improve training stability.

The most critical step in the training process involves hyperparameter tuning. We use a dropout rate of 0.1 subsequent to a fully connected layer to help avoid overfitting. In this case, we have used a batch size of 32, as a trade-off between memory constraints and convergence speed. Although the initial learning rate is 5e-4, different learning rates were tried to find the best one. We implement early stopping to avoid overfitting with a patience of 10 epochs such that if the validation accuracy does not improve, then training should stop. Because of these combined methods, the classifier performed better in correctly distinguishing between different types of LLMs.

5 Experiments

5.1 Experimental Setup

The experiments are conducted in a machine learning environment equipped with NVIDIA GPUs to leverage accelerated training times. Specifically, an NVIDIA GeForce MX 450 and Google Collab are used, allowing for the efficient processing of large batch sizes and deep learning models like BERT. The software environment consisted of the following key libraries:

1. PyTorch: Version 1.10.0 was utilized for building and training the neural network models.
2. Hugging Face Transformers: Version 4.11.3 was used to access pre-trained BERT models and tokenizers.
3. scikit-learn: For data preprocessing and evaluation metrics, including train-test splitting and classification metrics.

5.2 Evaluation Metrics

To evaluate the model's performance, several metrics like Accuracy, Precision, Recall, and F1-Score were employed.

Test Accuracy: 0.6103

	precision	recall	f1-score	support
gpt-2	0.55	0.73	0.62	876
gpt-neo	0.56	0.51	0.53	867
falcon	0.65	0.51	0.57	843
facebook-opt	0.72	0.68	0.70	917
accuracy			0.61	3503
macro avg	0.62	0.61	0.61	3503
weighted avg	0.62	0.61	0.61	3503

Figure 2: Metrics comparision for different LLM models

```
Epoch 1/3, Loss: 0.9936745762825012
Epoch 2/3, Loss: 0.991081178188324
Epoch 3/3, Loss: 0.9909428358078003
```

Figure 3: Epoch Vs Loss

5.3 Hyperparameters

The hyperparameters for training the classifier are carefully chosen to get the best performance:

1. Learning Rate: We set this to 5e-4 after some

initial testing showed that it helped the model learn effectively without causing instability.

2. Batch Size: A size of 32 is used, striking a good balance between memory usage and computational speed.

3. Dropout Rate: We added a 0.1 dropout after the first fully connected layer to help prevent overfitting and make the model more general.

4. Number of Epochs: The model was trained for up to 3 epochs, and we used early stopping to avoid overfitting when epochs are increased.

5. Weight Decay: A weight decay of 1e-2 was included in the AdamW optimizer to encourage simpler models and reduce overfitting.

6 Results

6.1 Quantitative Results:

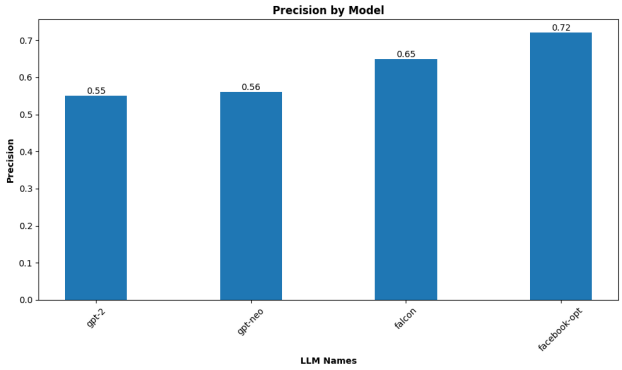

Figure 4: Precision for different LLMs

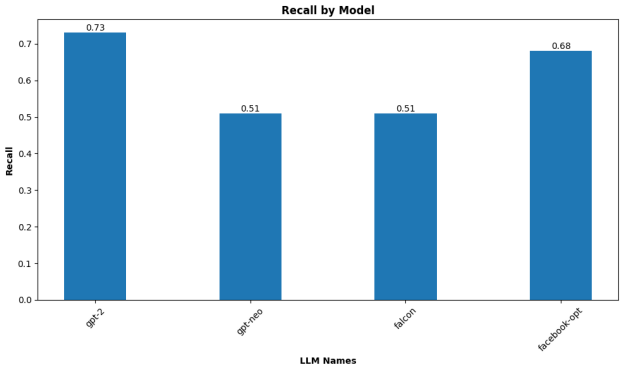

Figure 5: Recall for different LLMs

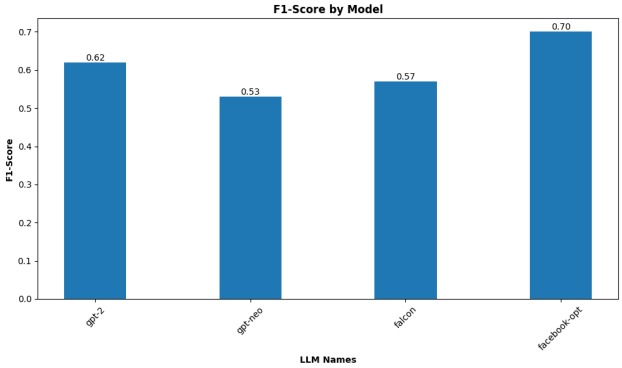

Figure 6: F1- score for different LLMs

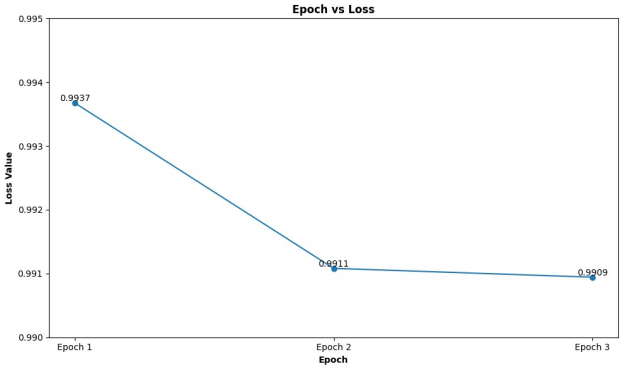

Figure 7: Epoch Vs Loss curve

6.2 Qualitative Analysis:

In this section, we present a qualitative analysis of the model's predictions by reviewing examples of text pairs and assessing how well it classifies the Large Language Model (LLM) that generated them.

Example 1:

- **Input Pair:**
  xi: "Yesterday I went"
  xj: "to Costco and purchased a floor cleaner."

- **Predicted LLM:** GPT-2
- **Discussion:** The model correctly identified GPT-2, as the text follows a typical narrative and structure seen in everyday language. The model's broad training data helped it recognize GPT-2's style.

Example 2:

- **Input Pair:**
  xi: "The findings of the study indicate"
  xj: "that there is a significant correlation between the two variables."
- **Predicted LLM:** Facebook-OPT
- **Discussion:** The model identified this formal, academic tone as characteristic of Facebook-OPT, which tends to generate structured, formal outputs.

The analysis shows that while the model performs well when LLMs have distinct styles, it struggles with more generic outputs. Expanding the dataset with diverse examples could improve its accuracy, especially in ambiguous cases.

7 Discussion of Related work

7.1 Interpretation of Results:

This work investigates the effectiveness of a deep learning classifier in differentiating several LLMs based on their complementing texts. Drawing upon contextual embeddings from a pre-trained BERT model, the classifier was able to pick up subtle variations in language style and syntactic patterns characteristic of each LLM.

It was particularly good at detecting distinctive linguistic styles, such as academic or technical prose, and tended to associate these strongly with specific models, such as Facebook-OPT or Falcon. It performed less well when the responses were more general or simple since these tended to sound very similar for different models. It underlines not only the complexity of the job but also how robustly the classifier can make use of the context in order to make an exact prediction even when there are subtle linguistic variations.

7.2 Error Analysis:

The analysis of model error showed varied misclassifications due to large generic responses. For example, statements as simple as "The weather today is sunny" were incorrectly given to specific models, since they lacked unique stylistic features that can only point to one model. There were also misclassifications whenever the prompts didn't work to similar outputs for multiple models, such as factual statements. These points use of more detailed prompts that can bring out distinct responses from each model.

7.3 Limitations:

The result is that again, deep learning classifiers can detect LLMs, but with one or two catches. First, the dataset of 35028 pairs of texts may be far too small to capture all the variety in the LLMs outputs. Increasing the number of samples in the dataset should increase performance.

Second, only a few LLMs were tested, which reduces the generalizability. Including more models might provide additional variations in which the classifier learns to tell the differences.

Finally, it was not easy to make out many models that were quite similar because most of the LLMs share characteristics and often are similar when their training data is related. That provides a clue that better design in prompting might help to highlight differences among the models. While the classifier

promises much, overcoming these limitations would be key to its improvement of accuracy for domains that are more practical.

8 Conclusion

8.1 Summary of Findings:

The study showed that the deep learning classifier can tell which LLM generated a completion of the text. This was enabled through the use of contextual embeddings derived from the pre-trained BERT model to detect subtle language patterns that distinguish LLMs apart. The model learned on a diversified dataset with pairs of texts and yielded good accuracy. On the flip side, the model fared poorly for non-specific completions and similar outputs across models, which can be highlighted as avenues for improvement.

8.2 Contributions:

Several key contributions are made by developing a whole new deep learning approach to classify LLMs depending on the unique features of their text completions. This paper creates a complete dataset of outputs from various state-of-the-art LLMs, laying the foundation for future research into how to attribute and study different models. It also contributes to ongoing work on NLP, since BERT embeddings could be combined together with a strong classification framework.

8.3 Future work:

The future direction for this work could be increasing the dataset to involve a wider variety of LLMs. This provides a better generalization of the results and performance of the classifier. Another strand of research involves the interpretability of the classifier. It may provide valuable views on the particular features featured in making the classifier distinguish between models. More complex architectures, for instance, making the classification process attended or employing transformer encoders, may further give the model improved discriminative capabilities. Advanced prompt engineering could result in richer and more distinctive varieties of output, thus better training the classifier on discerning between LLMs. While this work indeed established a sound basis for the text completion-based LLM identification, there is still much room for further improvements and strengthening of the classification systems in the future.

9 References

Here are some reference papers related to identifying LLMs based on completions:

1. Radford et al. discuss the multitask capabilities of unsupervised language models in their seminal work on GPT-2 [**radford2019language**].

2. Zellers et al. [**zellers2019defending**] introduce methods for defending against neural fake news, which could be relevant for differentiating between LLM-generated texts.

3. Clark et al. [**clark2021human**] present techniques for evaluating human assessments of machine-generated text, offering insights into how different LLMs' outputs may be compared.

4. Solaiman et al. [**solaiman2019release**] discuss the social impacts of releasing large language models, highlighting the importance of understanding their completions.

5. Gebru et al. [**gebru2021datasheets**] emphasize transparency in datasets, which is crucial when evaluating and classifying outputs from various LLMs.