LLM - Detect AI-Generated Text

AKSHAT SHAW
B.TECH. SOPHOMORE MMED, IIT-ROORKEE
OPEN PROJECT FOR VLG

1 Introduction

This project centres around the identification and classification of AI-generated text is a critical task in combating the proliferation of machine-generated content. The code within this repository showcases the process of building machine learning models to discern and classify text data generated by artificial intelligence systems. Due to massive advancements in the LLM and its capabilities, people often tend to generate their essays and academic works with LLM, so it becomes very important to identify the A.I.-generated text so that people doing their original work can be acknowledged.

2 The Science of Detecting LLM-Generated Texts

I have used this research paper on the Detection of LLM-generated texts for this section.

- Existing detection methods can be roughly grouped into two categories: **Black-box** detection and **White-box** detection. In practice, black-box detectors are commonly constructed by individual developers, whereas LLM developers generally carry out white-box detection.
- Black-box detection methods are limited to **API-level** access to LLMs. They rely on collecting text samples from **human and machine sources** to train a classification model that can be used to discriminate between LLM- and human-generated texts.
- To develop a proficient detector, black-box approaches necessitate gathering text samples originating from both human and machine-generated sources. Subsequently, a **classifier** is then designed to distinguish between the two categories by identifying and leveraging relevant features.
- As LLMs evolve and improve, black-box methods are becoming less effective. An alternative is **White-box detection**; in this scenario, the detector has full access to the LLMs and can control the model's generation behaviour for traceability purposes.
- For this project/task on Kaggle, we will use the **Black-box method**, as the dataset contains human and AI-generated text. Below, I have discussed what is the flow of work in the Black-box method **from scratch** without using the BERT pre-trained models.

2.1 Patterns in Human and AI-generated text

- Compared to human-authored text, which usually employs punctuation and grammar to convey subjective feelings—human authors frequently use exclamation points, question marks, and ellipsis to express their emotions—observations show that LLM-generated texts are less objective and emotional. LLMs likewise generate more formal and structured responses.
- Studies have indicated that human-written papers are more cohesive at the sentence level than texts created by LLM, which frequently reuses phrases in a paragraph.
- Research conducted on ChatGPT has revealed that texts written by humans typically have a shorter word count but a wider variety of language. The part-of-speech analysis underlines the dominance of nouns in ChatGPT texts, implying argumentativeness and objectivity, while the dependency parsing analysis demonstrates that ChatGPT texts use more determiners, conjunctions, and auxiliary relations.
- Sentiment analysis gives an indication of the text's emotional tone and mood. Unlike people, LLMs typically lack emotional expressiveness and are neutral by default.

- It is important to note that LLMs can substantially alter their linguistic patterns in response to prompts. For instance, incorporating a prompt like "Please respond with humour" can change the sentiment and style of the LLM's response, impacting the robustness of linguistic patterns.

During training, LLMs tend to depend on likelihood maximization objectives, which can create inconsistent or nonsensical text—a condition known as **Hallucination**. This highlights the importance of **fact-checking** as a critical component of detection. For example, it has been reported that OpenAI's ChatGPT **posts inaccurate news opinions** and produces **fake scientific abstracts**.

3 Data

- In the given dataset, we were provided with the essays generated by HUMAN and AI. We have to label the essays as either human- or AI-generated. The dataset given to us in this competition is highly skewed towards one label. The original dataset only contains 3 LLM-generated text and 1375 human-generated samples.

- To tackle this problem, we have to use an external dataset, or we can use API to generate essays by LLMs. Thankfully, we don't have to do this step on our own; many have uploaded datasets containing balanced labels. Here is the link to the dataset that I have used Dataset.

- Collecting data through human effort can be time-consuming and financially impractical for larger datasets. An alternative strategy involves extracting text directly from human-authored sources, such as websites and scholarly articles.

4 Preprocessing

- Preprocessing is a very important step to perform before training any model based on the NLP tasks, as preprocessing makes the text free from useless words, such as **stopwords, punctuations, numeric digits, URLs, etc**., that are present in the data, while the data collection and can greatly affect the model's performance.

- For the preprocessing part, I have used the **SPACY** library and **NLTK** for The stemming part, as lemmatising from Spacy, takes a lot of time to compile. I removed the stopwords, digits, punctuations, and URLs from the raw text and then used the **NLTK** library to perform the Stemming.

- In all of my notebooks, in order to reduce the submission time while submitting, I have preprocessed the train data, saved it as a CSV file and then used it again to save time on preprocessing.

5 Word Embeddings

- Unfortunately, Machine learning models can't understand text, so we have to somehow convert the text to numbers to train a machine-learning model on it.

- We have various methods for this, such as **Bag of Words, Tf-Idf, and N-grams**. These methods are based on the frequency of the word in the sentence, while word embedding, such as **Word2Vec** from the **Gensim** library, is based on deep learning techniques such as **CBOW** and **Skip-gram**, which use neural networks and create a vector-based representation for every word in the vocabulary.

- The Word2Vec-based techniques are very useful as they help identify the word vectors with proper context and perform best in NLP-related tasks.

- I have used every method for this project, and the Tf-Idf seems to perform better than the others. Also, I have used the **Byte-Pair** encoding from the hugging face library, along with the **Tf-Idf vectorizer**, which gave me the best **ROC-AUC** score for this project.

6 Model Building

- For the model-building part, I have observed that the **tree-based** algorithms perform much better than the linear model in the case of **NLP classification**. Also, the **Naive Bayes** algorithms perform much better than the linear models.
- In the given notebook in my GitHub, I have used many different models and methods like **cross-validation** and **GridSearchCV** to tune the **hyperparameters**.
- Further, I have a separate notebook in which I have used a **Bi-Directional LSTM** for which the preprocessing step is the same, but the word embedding technique for the **LSTM** is implemented in the model-building step itself.
- **LSTM** stands for Long Short-Term Memory, and these models are advanced versions of **RNN**-based networks which are specifically designed to address the **vanishing gradient** problem in traditional RNNs; LSTM layers have **memory cells**, which allows the network to remember or forget information over long periods selectively. This can be useful in **NLP** as context is required to be remembered within a sentence.
- However, the **LSTM** one does not perform better in this dataset. Finally, I used an ensemble model with models **MultinomialNB**, **SGDClassifier**, **LGBMClassifier**, and **CatBoostClassifier**.

I would like to thank **Zulqarnain Ali** as I have used his Notebook to improve my score and learned about the **Byte-Pair** encoding and the model's hyperparameters.

7 Results

- My final best score in this competition is **0.874**. This competition has ROC-AUC as the scoring matrices i.e., the Area under the curve of the **ROC** curve.
- Also, one notable strange thing happening in this competition is that people have a very good CV score, but their leaderboard score is not as good as the CV, and I think that this is not due to the overfitting but it is due to the hidden test dataset which is a very different text compared to the train data; this is also the reason for the poor performance of the LSTMs.
- I think that the host of the competition has degraded the quality of the AI-generated text in the hidden test set, i.e., it has more linguistic similarity to written human text. To tackle this problem, we need a dataset with text samples generated by LLMs, which are more similar to human style; thus, training on this data can improve performance.

8 Conclusion

In Conclusion, I found that in this competition, due to the hidden test data, simpler methods like TF-IDF and Word2Vec tend to perform much better as compared to the LSTM-based models since much of the details of the writing style of Humans and AI are different in train and test datasets.

Additionally, a few of the things that can be done to improve the score are as follows:-

- I think the test data has been tailored to be more similar to the human style of writing, as discussed in the above sections, so training the models with data that is more similar to humans but is LLM generated can help improve performance.
- Also, there are more grammatical mistakes in human written text as compared to AI, but the test data seems to have more grammatical mistakes, which the Host has purposefully done.
- Using advanced techniques like BERT-pre trained can improve the score as it is well-trained.
- Various other models can be tried with better hyperparameters, which can slightly improve the score.

9 Extras

I have deployed this project end-to-end on StreamLit using its frontend, and it is online. Click here! To use the app. Also, here is the GitHub repository for the same.

10 Reference

Tang, Ruixiang, Yu Chuang, and Xia Hu. "The Science of Detecting LLM-Generated Texts." ArXiv, (2023). Accessed January 14, 2024. /abs/2303.07205.