OPEN PROJECT REPORT

LLM-detect AI Generated Text

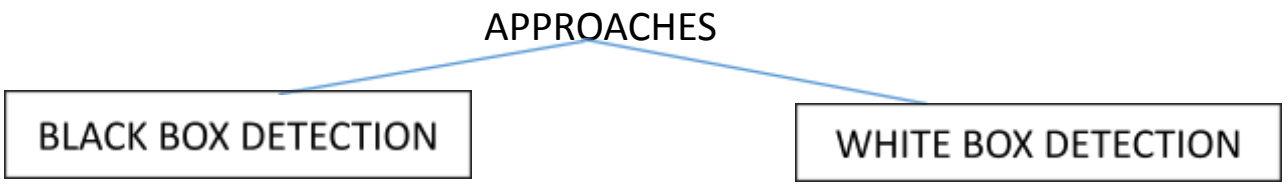
KRITI GARG|METALLURGICAL AND MATERIALS ENGINEERING

Analysis of the Research done till now

● INTRODUCTION :

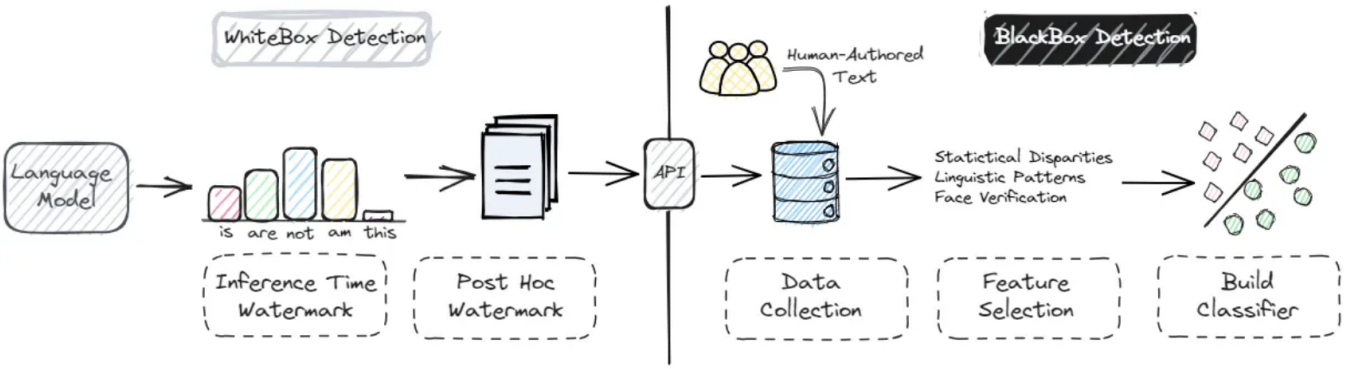
With the advent of LLMs such as Chat GPT, Bard it has become significantly easy to plagiarize texts,spread false information and they have the ability to effectively replicate human speech. Therefore, to maintain ingenuity, it becomes important to detect LLM generated text.
For this we have multiple approaches. The approaches discussed are:
(1) Traditional Classification Algorithms
(2) Deep Learning approaches

These approaches can be divided into two broader groups:



Black Box Detection Methods:. Black-box detection methods are limited to API-level access to LLMs. They rely on collecting text samples from human and machine sources, respectively, to train a classification model that can be used to discriminate between LLM- and human-generated texts.Black-box detectors work well because current LLM-generated texts often show linguistic or statistical patterns.
Subsequently, a classifier is then designed to distinguish between the two categories by identifying and leveraging relevant features.

White Box Detection Methods:The detector has full access to the LLMs and can control the model's generation behavior for traceability purposes. In practice, black-box detectors are commonly constructed by external entities, whereas white-box detection is generally carried out by LLM developers.



● Black Box Detection

The basis of this is to perform binary classification.The aim is to capture textual features that differentiate human generated from LLM generated texts.

Let's have a look at the Classification Models:

(1) Traditional Classification Algorithms: Algorithmssuch as SVMs, Naive Bayes,Decision Trees, Random Forests.. One advantage of these algorithms is their interpretability, allowing researchers to analyze the importance of input features and understand why the model classifies texts as LLM-generated or not.

(2) Deep- Learning Approaches: This involves fine tuning pre-trained models such as BERT, RoBERTa. We use these pre-trained models as foundation.

Black box detection with unknown Model sources:In such cases, teachers are often unaware of the specific AI service being employed. Consequently, this situation poses the greatest challenge as very limited information is available to identify instances of deception.

Black box detection with known Model sources:We possess knowledge regarding the specific model from which the text originates, yet we lack access to its underlying parameters.

● White Box Detection

In white-box detection, the detector processes complete access to the target language model, facilitating the integration of concealed watermarks into its outputs to monitor suspicious or unauthorized activities.

Types of watermarking:

Post-hoc watermarking:Post-hoc watermarks insert a hidden message into LLM-generated text for later verification. To check for the watermark, one can extract this hidden message from the text in question. These watermarking methods primarily fall into two categories: rule-based and neural-based.

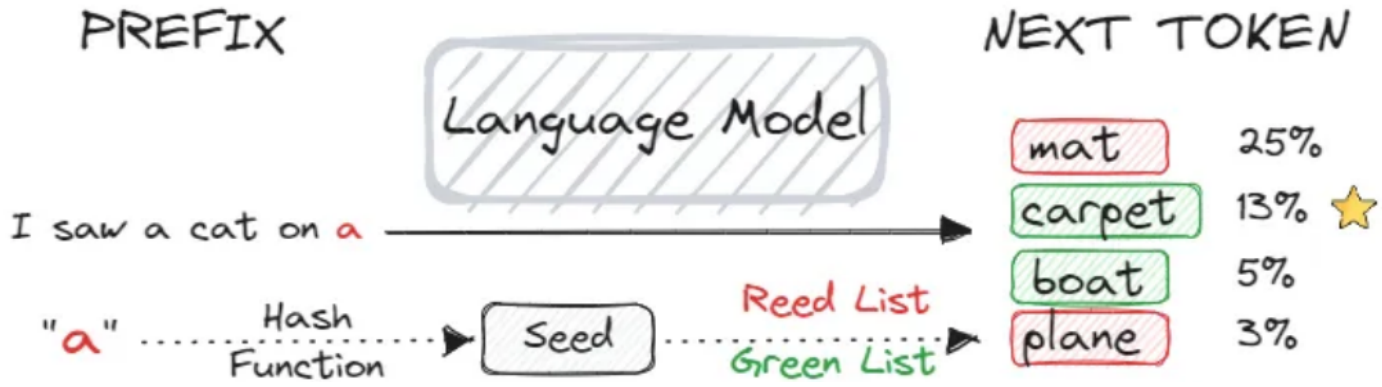
Inference Time Watermarking:This method alters the word selection during a decoding phase. The model creates a probability distribution for the next word, and embeds a watermark by tweaking this selection process. Specifically, a hash code generated from a previous token categorizes vocabulary into *"green list"* and *"red list"* words, with the next token chosen from the green list.

White-Box Detection with Full Model Parameters:

When we have full access to the model, GLTR trains a logistic regression over absolute word ranks in each decoding step.

White-Box Detection with Partial Model Information:

When only partial information like the model output logits are available, SeqXGPT introduce a sentence-level detection challenge by synthesizing a dataset that contains documents that are polished with LLMs and propose to detect it with logits as waves from white-box LLMs.



Black-Box Watermarking

It first defines a binary encoding function to compute a random binary encoding cor responding to a word. The encodings computed for non-watermarked text conform to a Bernoulli distribution, wherein the probability of a word representing bit-1 is approximately 0.5. To inject a watermark, they alter the distribution by selectively replacing words representing bit-0 with contextbased synonyms that represent bit-1. A statistical test is then used to identify the watermark.

- **Limitations of Black Box Detection:**

(a) **Bias in collected datasets**:Data collection plays a vital role in the development of black-box detectors, as these systems rely on the data they are trained on to learn how to

identify detection signals.Data collection plays a vital role in the development of black-box detectors, as these systems rely on the data they are trained on to learn how to identify detection signals.

(b) **Confidence Calibration**:. In other words, the predicted class probabilities should reflect its ground truth correctness likelihood. Accurate confidence scores are of paramount importance for assessing system trustworthiness, as they offer valuable information for users to establish trust in the system, particularly for neural networks whose decisions can be challenging to interpret.

● Limitations of White Box Detection:

(a) There exists a **trade-off between the effectiveness of the watermark and the quality of the text**, which applies to both post-hoc watermarks and inference time watermarks. Achieving a more reliable watermark often requires significant modifications to the original text, potentially compromising its quality.

(b) When attackers query the LLMs, they can gain knowledge about the green list words proposed in the inference time watermark . This enables them to **generate text that fools the detection system.**

● Zero Shot Methods:

The following zero-shot methods are used as baseline models:

- **Log Likelihood**: First, this method uses an LLM to calculate the probability of each token xi in the text X = (x1, x2,. . . ,xt), then uses the average log-probability of each token in the text as a statistical feature. This value is larger, and the probability that the detected text is LLM-generated text is higher.

- **Log Rank**: According to the probability distribution of each token in detected text X = (x1, x2,. . . ,xt), the absolute rank of each token can be calculated. This method uses the average log-rank of each token in the detected text as a statistical feature. This value is smaller, and the probability that the detected text is LLM-generated text is higher.

- **Entropy**: The average entropy of the probability distribution of each token in the detected text is used as the statistical feature. The smaller this value is, the higher the probability.

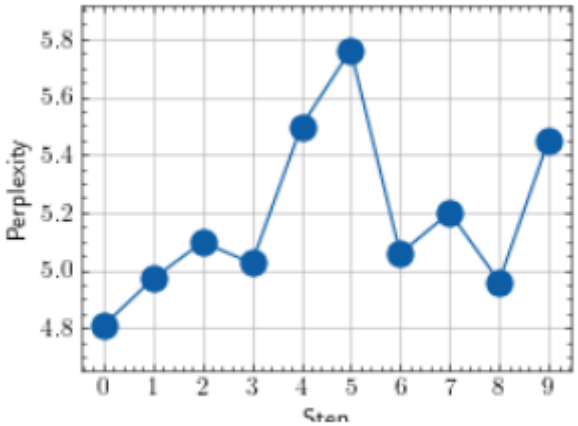
- **GLTR**: The rank of each token in the detected text is first obtained. Then all tokens in the detected text are divided into four groups according to their rank with 10, 100, and 1000 as boundaries. Finally, the percentage of the number of tokens contained in these four groups is taken as a set of statistical features. This set of statistical features is used as input to a classifier, which outputs the probability that the detected text is a text generated by LLMs.

- **DetectGPT**: Firstly, the perturbation model generates several perturbed texts of the detected text. The average difference between the log probabilities of the original text and several perturbed texts is then measured and perceived as a statistical feature.

- **DetectLLM-LRR**: In this method, the statistical feature is the Log-Likelihood Log-Rank Ratio of the detected text.

● What Are Some Metrics used by AI Content Detection Tools?

(1) **Perplexity:** It is a measure of newness of text to the model. Higher perplexity means the text is generated by LLM. Lower perplexity means that the text has been generated by student.
A score of 0 indicates perfect predictability, meaning that the next word is always the most likely word. A score of 100 indicates the worst possible predictability, meaning that the next word is never the most likely word.
Human-generated text typically has a perplexity score of around 20-30. This means that human-generated text is relatively predictable, but it is not completely predictable.
AI-generated text, on the other hand, typically has a perplexity score of around 50-100. This means that AI-generated text is much less predictable than human-generated text.




```
>>> # Example plotting a single value
>>> import torch
>>> from torchmetrics.text import Perplexity
>>> metric = Perplexity()
>>> metric.update(torch.rand(2, 8, 5), torch.randint(5, (2,
8)))
>>> fig_, ax_ = metric.plot()
```

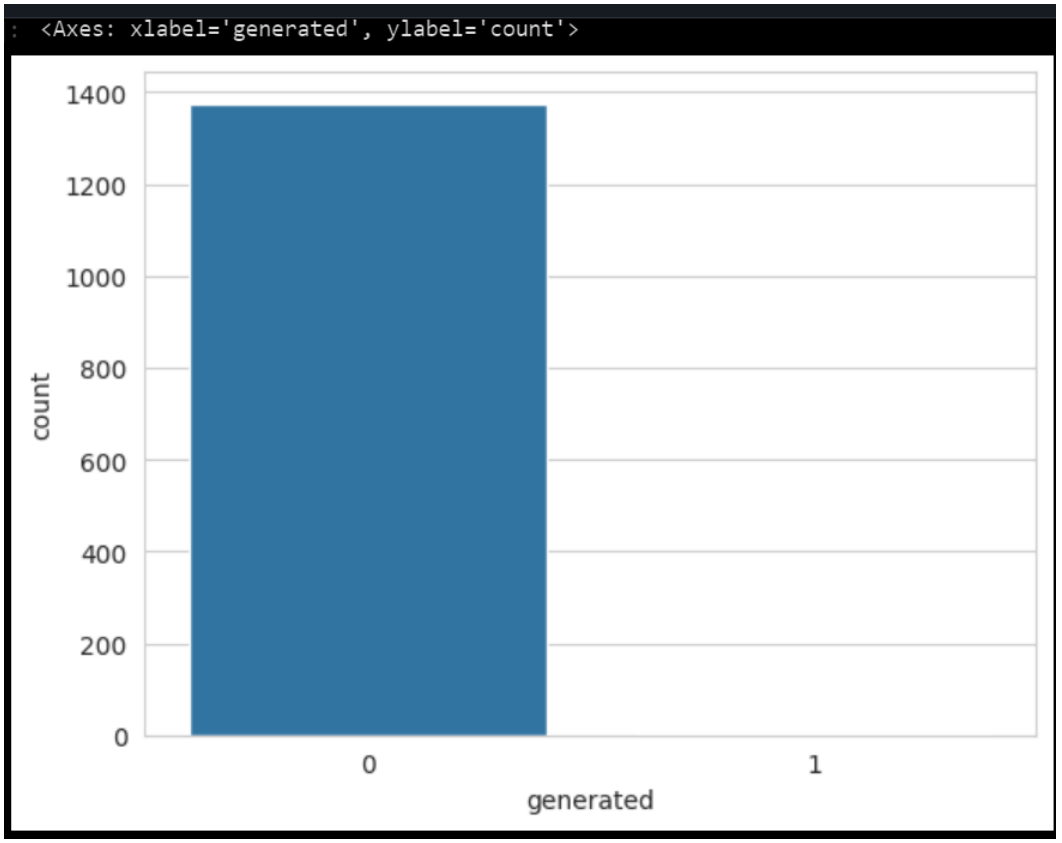
(2) **BURSTINESS:**Burstiness refers to the distribution of topics or ideas within a piece of content. It measures how frequently certain topics or ideas appear in a short span of time. A bursty content has frequent occurrences of certain topics, resulting in a non-uniform distribution.

Burstiness measures the variability of the sentence lengths in a text.

PROCEDURE:

(1) DATA COLLECTION:

Data has been collected from kaggle datasets. The links to them are given below:
https://www.kaggle.com/competitions/llm-detect-ai-generated-text/data
Extra data was added to handle imbalanced data.



(2) **EXPLORATORY DATA ANALYSIS:** Data has been analyzed in this step. The missing values are handled. Data is visualized using matplotlib.pylot library. 'Generated'=0 refers to student written essays while 'Generated'=1 refers to LLM generated essays. The percentage of student generated/LLM generated is visualized in the form of a pie chart.

```
labels=["Student","LLM"]
student_or_not=train_data['generated'].value_counts().tolist()
values=[student_or_not[0],student_or_not[1]]
fig=px.pie(values=train_data['generated'].value_counts(),names=labels,width=500,height=500,color_discrete_s
fig.show()
```

(3) MODEL TRAINING: As discussed above, model can be trained using classification algorithms such as SVM, Naive Bayes Algorithm. Also aother approach is to fine tune the model using pre-trained network such as BERT, GPT-3,RoBERTa.
Following libraries will be imported for Naive Bayes Algorithm.

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
```

(4) MODEL EVALUATION: Model will be evaluated using accuracy score, F1 Score, confidence score. Naive Bayes is giving an accuracy score of 0.9806 on the testing set.

```
accuracy=accuracy_score(y_test,y_pred)
accuracy
```

```
0.9806657364959139
```

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.98      3051
           1       0.99      0.96      0.97      1966

    accuracy                           0.98      5017
   macro avg       0.98      0.98      0.98      5017
weighted avg       0.98      0.98      0.98      5017
```

● WORK TO BE DONE POST MID EVAL:

(1) Using other Classification Algorithms such as SVM, Random Forests with Logistic Regression as the baseline.
(2) Fine tuning pre-trained models such as BERT , RoBERTa.

● RESOURCES USED:

✔ https://arxiv.org/pdf/2310.14724.pdf

✔ https://arxiv.org/pdf/2310.15654.pdf

✔ https://assets.researchsquare.com/files/rs-3226684/v1_covered_def36854-d747-4cc4-b19a-7d3b559405d3.pdf?c=1693806752

✔ https://towardsdatascience.com/challenges-of-detecting-ai-generated-text-6d85bf779448

✔ https://arxiv.org/ftp/arxiv/papers/2307/2307.12166.pdf

✔ https://arxiv.org/pdf/2305.18149v3.pdf

- ✔ https://arxiv.org/pdf/2305.10847v5.pdf
- ✔ https://arxiv.org/pdf/2310.18906.pdf
- ✔ https://miamicloud.com/ai-detectors-exposed-everything-you-need-to-know-about-llm-ai-generated-text-unveiled/
- ✔ https://betterprogramming.pub/detecting-llm-generated-texts-befce4426da9