Authenticity Master: Machine-Generated Text Detection Utility using Bidirectional Encoder Representations from Transformers (BERT)

Jan Bermudez[1] **Francisco Cilia**[1] **José Renteria**[1]

Abstract

The growing proficiency and widespread use of large language models for generating text content highlights the imperative need for a corresponding tool to aid in detection of such machine generated text. As LLMs[2] continue to grow, it becomes increasingly difficult to detect if a piece of text is authentic or written by an AI model. This project aims to provide a utility to aid in providing an accurate report on whether text was generated by a LLM or not. It does so by utilizing a discriminate model which is a modified version of the BERT model for classification, and is tested on multiple model architectures. A more flexible architecture than the generative model (e.g. being bidirectional) makes it significantly more accurate for detection.

Introduction

Large Language Models (LLMs) have demonstrated the capability to provide highly fluent and comprehensive replies to a diverse range of user queries. Examples include LLM Models GPT-3 (Brown et al., 2020), PaLM (Chowdhery et al., 2022), and ChatGPT (OpenAI, 2022), which all exhibit the ability to produce answers to complex questions about several fields ranging from society, history, mathematics and science. The widespread accessibility of LLMs presents a significant societal impact, touching upon several aspects of human interaction and cybersecurity. On one hand, LLMs clearly offer an unprecedented opportunity for innovation, facilitating advancements in Natural Language Processing, content generation, and communication. However, this accessibility also raises questions regarding the integrity and authenticity of information that is being generated. Given the potential for LLMs to generate convincing content that is potentially fabricated or inaccurate, there arises a critical need for a robust mechanism to verify information authenticity and combat the rapid increase of the spread of this information. The democratization of LLMs may exacerbate cybersecurity risks, as malicious attackers could exploit these models to generate sophisticated phishing emails, convincing fraudulent messages, or deceptive content that aims at manipulating the public opinion or compromising sensitive data. This is the reason we decided to pursue this project, as we recognized the importance and urgent need of implementing a stringent utility that is able to quantify and measure the authenticity and potential threat that lies within an email received or just any body of text you may deem suspicious. Our utility also provides a tool that can predict the risk of an email sender. This is achieved through the integration of a system that uses hundreds of factors such as domain age, traffic rankings, presence on other sites, public records, deliverability, data breaches, phishing emails and more to provide a general idea about whether the email you received is risky or comes from a trustworthy sender. This utility provides a measure to aid in detecting email authenticity and potentially risky senders.

[1] University of Oregon, School of Computer and Data Sciences, Department of Computer Science
[2] Large Language Models

Keywords

Artificial Intelligence Detection Tools, Originality.ai, Bidirectional Encoder from Transformers, Email Reputation

Related Work

When looking at related work, we review work that proposes a different method to solve the same problem or work that uses the same proposed method to solve a different problem.We may also examine similar methods that solve a similar problem and related problems that fall within our problem domain.

Currently, there exist some tools that can help users detect AI generated content. These include Content at Scale, Originality.ai, Writer AI, and ZeroGPT. All these tools provide AI generated content using different and similar methods. Our chosen provider, Originality.ai uses the Bidirectional Encoder from Transformers model for classification but there are many others.

This class of statistical language models is able to predict the next word in a sequence given the words that precede it. Other models include:

- **N-Gram:** This method is arguably the simplest approach to language modeling. It involves establishing a probability distribution for a sequence of 'n' elements for any number n, where 'n' defines the size of the gram (a sequence of words assigned a probability). For example, if n equals 4, a gram might appear as "The car is red" and signifies the level of context the model is instructed to take into consideration. N-Gram models encompass numerous types including unigrams, bigrams, and trigrams.
- **Unigram**: The unigram can be the most basic form of language model as it does not consider conditioning context in its computations. It assesses each word or term in isolation and independently. Unigram models often find usage in language processing tasks such as information retrieval. Serving as the cornerstone for a more specific model variant known as the query likelihood model, the unigram employs information retrieval to inspect a collection of documents and identify the most relevant one corresponding to a particular query.
- **Continuous Space:** This type of model. words are organized as a non-linear combination of weights within a neural network structure. It uses a process known as 'embedding' where each word is assigned a weight. This model has been found to be useful in situations where the dataset is filled with an extensive vocabulary with many unique words and expands significantly. In such cases where the dataset is large and unique, linear models like n-gram are limited in their ability. This limitation is due to the fact that as the number of unique words increases, the number of possible word sequences increases as well, leading to weak predictive patterns.
- **Exponential**: Another type of statistical model examines text through an equation that integrates n-gram and feature functions. This works by leveraging predefined features and parameters for the decided outcomes. These features include various linguistic attributes or characteristics of text. Parameters are adjustable components within the model and influence and its behavior and performance. This model is based on the principle of entropy, where the model decides that the probability distribution with the most entropy is the best choice. Exponential models entail fewer statistical assumptions and therefore increasing the likelihood of accurate results.

As discussed, there exists several methods that are used to detect AI generated content, all with their

[3]S

own strengths and weaknesses. Originality.ai uses the Bidirectional Encoder from Transformers model for classification and it is primarily employed for our program. Language models are useful for a wide variety of tasks that include but are not limited to speech recognition, grammar induction, information retrieval, optical character recognition and natural language generation. This paper acknowledges the extensive field of language models and it is important to note the current scope does not encompass other classes of models such as neural language models. Analysis of threat models indicates that when utilized correctly, machine-generated text detection is a valuable tool for reducing the negative impacts of NLG model abuse. While there exists numerous sites and programs for detecting AI generated content, not all offer a comprehensive suite of security features. Our program builds upon these foundations and offers a comprehensive solution for users in a single place.

Design, Algorithm, Implementation

When designing our program, we knew we needed a way to quantify and measure the authenticity and potential threats within a received email or plain text we deem suspicious. At its core, we wanted the program to utilize several APIs to access pre-trained language models to create a seamless all encompassing security experience for the user. This set the stage for the first step in finding the AI detection providers that would allow us to provide our users with AI content detection through both plain text and URL, and a service to determine the risk associated with an email address.

When evaluating AI detection API providers, it is crucial to determine several factors such as accuracy of the detection (false positives, false negatives), the range of features supported, pricing, ease of integration and accompanying documentation. We assessed numerous options including Sapling.ai, CopyLeaks, GPTZero, Writer, and Originality.ai.

All of these given options allow developers to send in text through an API call, have it analyzed and then send a response with a probability score. The score would indicate how confident the provider is that the text was either AI or human written. With the wanted functionally being similar among them all it came down to accuracy and the pricing.

1. ***Sapling.ai*** offers their service for $25 a month for their pro plan designed for individuals
2. ***CopyLeaks*** offers their service for $13.99 a month and covers 300,000 scanned words (1,200 credits).
3. ***Originality.ai*** has their service at $14.95 a month and includes 2000 credits or 200,000 words
4. ***GPTZero*** starts their API plan at $45 a month for 300,000 words
5. ***Writer*** starts their individual plan at $18 per month
6. ***Originality.ai*** offers AI content detection through both plain text and URL. Their response contains the overall prediction score as well as for subsections of the text.

Name	Public Demo	Chrome Extension	API Endpoints	HTML support	Overall Score	Sentence Scores	Result sharing
Sapling.ai	Yes	Yes	Yes	Yes	Yes	Yes	Yes
CopyLeaks	Yes	Yes	Yes	No	Yes	Yes	No
Originality.ai	No	No	Yes	Yes	Yes	Yes	No
GPTZero	Yes	Yes	Yes	No	Yes	Yes	No
Writer	Yes	No	Yes	No	Yes	No	No

Ultimately, we decided to go with ***Originaliy.ai*** for their supportive documentation, full site scan from URL, scan history, file upload and accuracy. "Originality.AI Launches Version 3.0 Turbo (the most accurate AI detector ever) to work alongside version 2.0 Standard resulting in an improvement on the most challenging dataset created from the newest LLMs. Accuracy improved from 90.2% to 98.8% False Positives reduced from 2.9% to 2.8%"[4]

With this chosen AI detection provider, we only needed to figure out how to include a tool that can predict the risk of an email sender. For this solution we came across EmailRep.
"EmailRep uses hundreds of factors like domain age, traffic rankings, presence on social media sites, professional networking sites, personal connections, public records, deliverability, data breaches, dark web credential leaks, phishing emails, threat actor emails, and more to answer these types of questions: Is this email risky? Is this a throwaway account? What kind of online presence does this email have? Is this a trustworthy user?" (4)

To accommodate users with varying preferences for accessing applications, we incorporated our API into two separate programs. A Python GUI[5] and a website using the React framework.
Overall, providing two different types of applications offers flexibility, accessibility and gives users a choice to ensure compatibility with a wide range of platforms and devices.

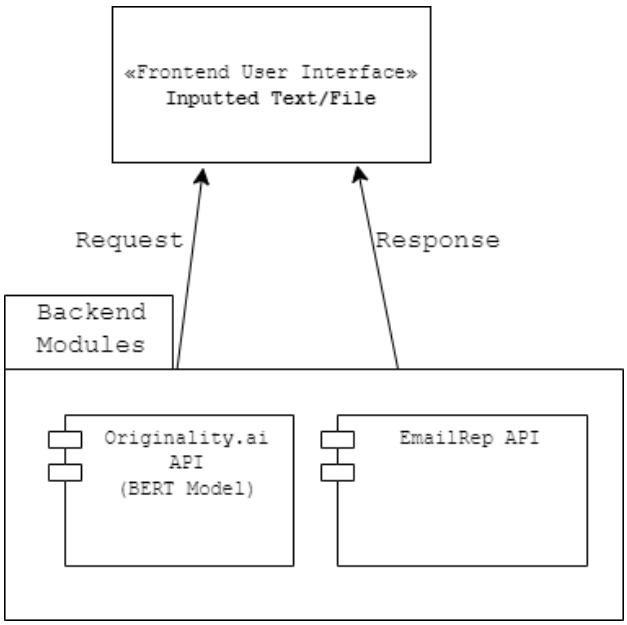
Python GUI Application

- A graphical user interface implemented using tkinter, matplotlib for data visualization, and requests for making http requests.
- app.py is the main file where the graphical user interface (GUI) and program logic is implemented.
    - It contains the code to create the geometry and configurations of the screen, frames, labels, input boxes, and buttons.

Additionally, it includes handlers to send API requests

- def ai_scan(self)
    - This function is responsible for initiating an AI detection scan to Originality.ai's API. Inputted text from user is gathered and placed as the payload
    - response = requests.request("POST", url, headers=headers, data=payload)
- def ai_url_scan(self)
    - Similarly, this function initiates a URL scan request to Originality.ai's API.
- def scan_email(self)
    - This function handles the email scanning request to emailrep.io's API
- controller.py is the entry point for instantiating the application and running it.

Software Architecture model for Python GUI Application, follows Client/Server Architecture.

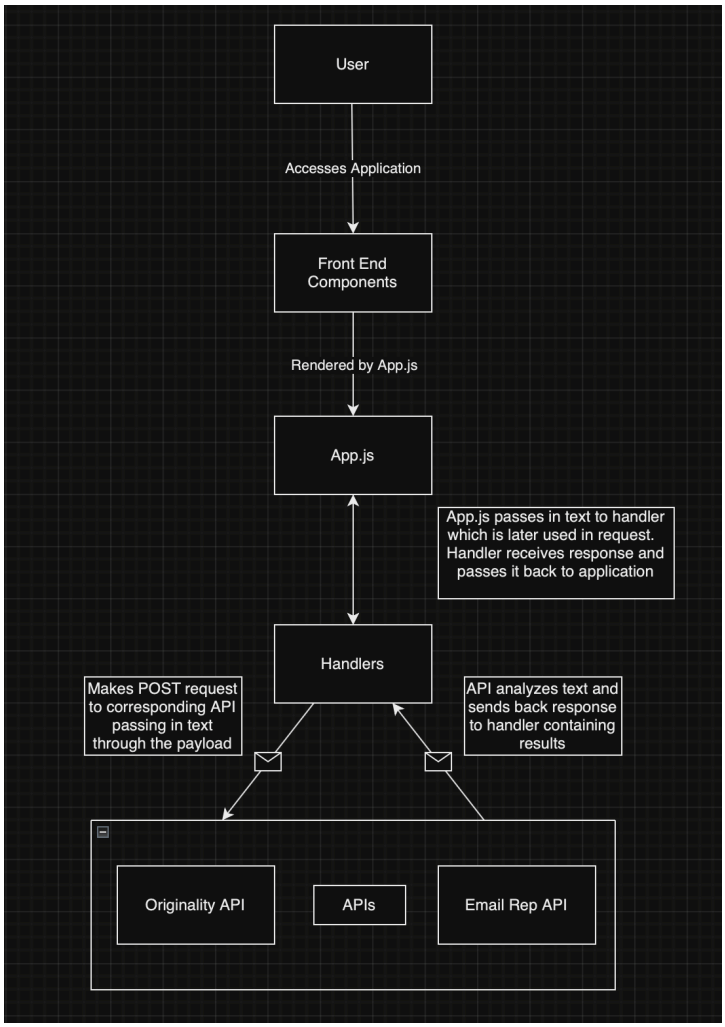


[4] Gillham 3
[5] Graphical User Interface

React Web Application

- A web-based application using the React framework to analyze text data entered by users with various components and handlers for the different API calls.
- The main component of the website is the landing page (App.js)
    - It contains a large text box centered in the middle for the user to enter either plain text, a URL or an email address.
    - Under the text box are two buttons for the user to press the corresponding action for the inputted text
    - The navigation bar with our program name “Authenticity Master”
    - A display component that renders box showing the results from the scanned text
- The handlers send POST requests to their respective API with the text to be analyzed. Although all the handlers are functional some were disabled to keep the scope of the project focused.
    - aiHandler.js
    - urlHandler.js
    - aiPlagirismHandlers.js
    - emailHandler.js
    - plagirismaHandler.js
    - loadingSpinner.js
- There are multiple CSS files to style the different components rendered by the application.

Software Architecture Model for React Web Application, which similarly also follows Client/Server architecture.

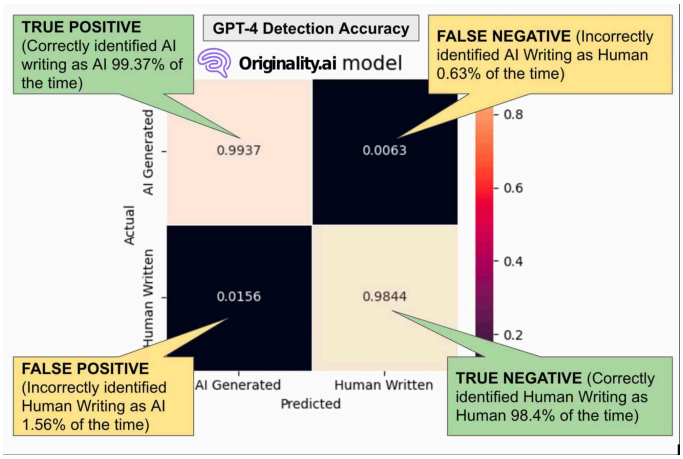


Client Server Model

The main reason that we chose to use this architecture for our project was because it allowed us to clearly separate the client-side and server-side components of the application. By doing this, we created a clear and maintainable division of responsibilities. In this case the client is responsible for handling the user interface and rendering the appropriate components, while the server manages the data processing. Additionally, adopting this architecture allows us to achieve better scalability in the future if we choose to pursue it. This approach allows the server to handle multiple requests concurrently which helps guarantee better performance under heavy traffic. It also provides us with an easy transition to adding more server resources or using load balancing techniques if our service is continuously under heavy stress.

Evaluation Methodology

By utilizing pre-trained models from trusted and professional sources, we ensured reliability and accuracy in our evaluation process. The following figure demonstrates the Originality.ai model's accuracy in detecting GPT-4 generated content.



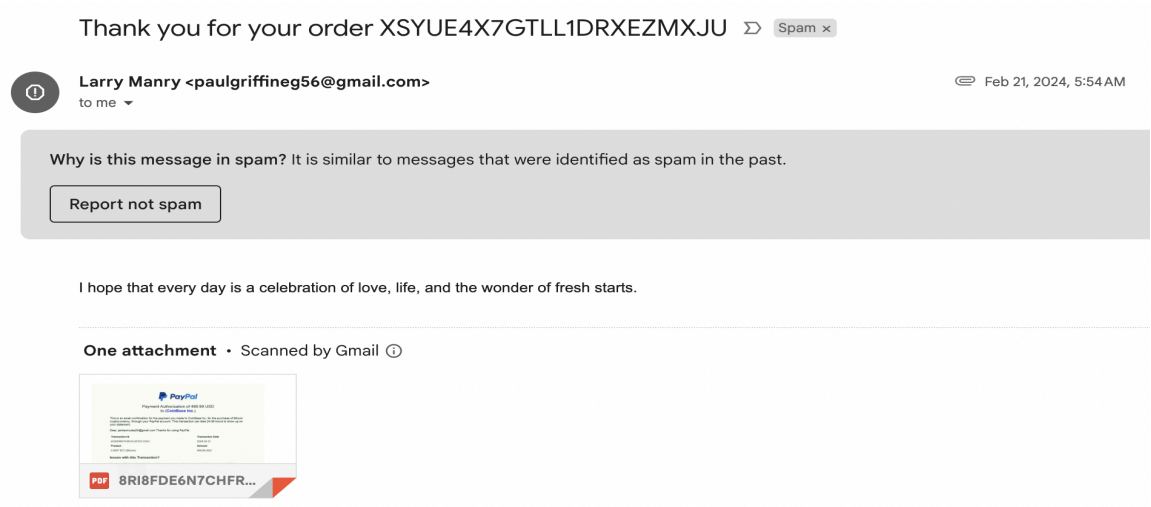
In summary, their model incorrectly identified AI generated writing as human 0.63% of the time and identified human writing as AI 1.56% of the time.

To test and assess the claims of their study, we conducted several tests by subjecting our program to an array of both AI generated and human generated content. This assessment involved prompting chatGPT to write various expository and narrative paragraphs. We asked chatGPT to generate paragraphs with prompts such as "write a 60 word paragraph about a dog" and "explain how to make a sandwich in 60 words".

For human generated content, we curated excerpts from personal course essays from previous terms and and published textbooks such as Herman Melville's *Moby-Dick; or, The Whale.* These sources were chosen to capture a broad spectrum of writing styles, subjects and levels of complexity. In testing for the email scanner, our process involved testing against our personal emails as well as emails from large corporations as a baseline for known harmless emails.

Additionally, we gathered real-world phishing emails that we had previously received, leveraging these instances we are able to assess the effectiveness of EmailRep from distinguishing legitimate and malicious emails.

As for the URL scanner, identifying websites containing AI-generated content required a simple google search to find existing repositories of known sites with AI generated content. We were able to identify a range of websites hosting AI generated content which allowed us to evaluate the performance of our URL scanner. We aimed to assess the programs performanting across a variety of writing styles and complexities to ensure a robust and accurate analysis to gauge its detection capabilities.



Results and Analysis

- Our program resulted in a suite of security features for the user, each with a high rate of detection and accuracy.

Discussion of Complicated or Unresolved Issues

In discussing complicated or unresolved issues, one interesting avenue for further research allows us to dive into the possibility of a generative adversarial network (GAN). “A generative adversarial network (GAN) is a class of machine learning frameworks and a prominent framework for approaching generative AI.[1][2] The concept was initially developed by Ian Goodfellow and his colleagues in June 2014.[3] In a GAN, two neural networks contest with each other in the form of a zero-sum game, where one agent's gain is another agent's loss.”
The core idea of a GAN is based on the concept of “indirect” training facilitated by a discriminator, an additional neural network that is tasked to gauge how “realistic” the input seems, which is continuously updated. Rather than training the generator to minimize its deviation from a target image, the objective is to deceive the discriminator. This approach allows the model to learn in an unsupervised manner as it strives to produce outputs that convincingly mimic authentic data.

Exploring the potential applications of GANs within our context of AI generated content detection presents a variety of opportunities and challenges. GANs could offer a powerful framework for generating realistic synthetic data that has the ability to enhance the diversity and complexity of test cases used to evaluate detection algorithms. By producing a broader spectrum of AI generated content, researchers could gain a better insight into the limitations of existing methods and ultimately drive improvements in detection accuracy and reliability.

However, the incorporation of GANs into content detection also could pose several issues. For example, ensuring ethical and responsible use of GAN generated content raises concerns about the potential misuse or exploitation, in particular the potential misuse or exploitation, in particular the context of creating deceptive or harmful content for malicious purposes. In this case, it would be making the jobs of applications such as AuthenticyMaster that much more difficult in distinguishing AI generated content from human authored work.

Despite these challenges, the exploration of GANs holds promise for advancing the field of AI generated content detection. With further discussion and research, the capabilities and limitations of GAN-based approaches can contribute to the development of more robust and effective methods for detection and ultimately enhancing cybersecurity and safeguarding against the proliferation of AI generated misinformation.

Lessons Learned

The process of training and tuning machine learning models for natural language processing (NLP) has been introduced to several lessons. Firstly, we encountered a steep learning curve in the intricacies of NLP model training. Moreover, we’ve learned the importance of acquiring and curating high-quality datasets and the time for training a language based model is far more intensive than for a model for binary classification and a few features. The considerable time and computational resources required for training a language-base model far exceed ours. Additionally, the tuning process involves an iterative approach with various hyperparameters and optimizations to arrive at an optional performance of detection. This iterative nature, together with the large datasets significantly extends training time with each iteration.

In summary, the trials of training and tuning machine learning models for NLP are marked by various challenges and valuable lessons that are outside the scope of this term. Moving forward, applying these insights will be crucial to meet milestones and overcome challenges in natural language processing.

Conclusions

As we continue to see advancements in Large Language Models, they are rapidly becoming appealing options for replacing human writers across diverse applications such as education, journalism, or even technical writing. While these technologies offer valid applications in these domains, there is a growing expectation from educators, readers, and consumers to have mechanisms in place for confirming authentic human authorship of content, especially when accuracy holds significant importance in a crucial context, such as cybersecurity.

Given the heightened significance of this issue and the frequent introduction of new large language models, we investigated the problem of detecting machine-generated text utilizing a Bidirectional Encoder Representations from Transformers or BERT model for short.

In this approach, we relied on the inherent log probabilities calculated by this pre-trained generative model to determine whether a given passage originated from a generative large language model or not. We utilized the log probability which is computed with a large data-set including samples from a wide variety of large language models, showing that a tractable approximation of the function provides a useful and quantifiable signal that a given sample was machine-generated.

A limitation of this probability-based approach for machine-generated text detection is the assumption that we can evaluate log probabilities of the model(s) in question. For model's behind API's which do not provide any insight into the probability such as OpenAI's GPT-3, evaluating such a probability would be costly.

While BERT models offer remarkable capability in understanding and analyzing language, they are not without limitations when it comes to LLM-generated text detection. One significant limitation lies in the adaptability of LLMs themselves.

As LLMs continue to evolve and undergo training on increasingly more vast datasets, they will continue to become more sophisticated and adept at generating text that more closely resembles human-generated content. This advancement poses a significant challenge for our current approach and other existing detection mechanisms, as LLMs may eventually be able to produce content that is indistinguishable from genuine human authored content. Consequently, the effectiveness of BERT models for LLM detection may diminish over time, potentially rendering them completely obsolete for this specific purpose. This highlights the need for future research and ongoing development of detection techniques that can keep pace with the rapid evolution of LLMs in order to effectively identify text generated by these models.

Future Work

While the methods in this work make an attempt to not assume anything about the models generating our samples, future work may explore how watermarking algorithms can be coupled with a detection model such as the one provided by *Originality.ai* to further improve their reproductions of human text. Future models may be able to use an aggregate score compiled by several trained models that can more accurately dictate whether a sample was generated using an LLM.

A topic we did not consider much but we know lies in the horizon is that of the relationship between prompting and detection; that being, can a cleverly crafted prompt successfully prevent a generated sample from being detected by any existing models? This could include various methods of tweaking the sample to rid itself of clear giveaways such as monotonicity or reuse of common language, including tone and perspective shift such as writing from the first or third person could also make it increasingly more difficult to detect via it sounding more humanesque. All in all, there still remains a lot to consider for this particular issue, as it will likely be a generation-defining problem now that artificial intelligence and LLMs specifically have become such an integral part of our daily lives.

References

"AI Content Detector Accuracy Review + Open Source Dataset and Research Tool – Originality.AI" originality.ai/blog/ai-content-detection-accuracy.

Bowman, Samuel R. "Eight Things to Know about Large Language Models" 4 Apr. 2023, arxiv.org/pdf/arxiv.org/pdf/2304.00612.pdf.

"DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature" 23 July 2023, arxiv.org/abs/2301.11305.

E. N. Crothers, N. Japkowicz and H. L. Viktor, "Machine-Generated Text: A Comprehensive Survey of Threat Models and Detection Methods," in IEEE Access, vol. 11, pp. 70977-71002, 2023, doi: 10.1109/ACCESS.2023.3294090.

"Generative Adversarial Imitation Learning" papers.nips.cc/paper_files/paper/2016/hash/cc7e2b878868cbae992d1fb743995d8f-Abstract.html.

Radford*, Alec. "Language Models are Unsupervised Multitask Learners" 14 Feb. 2019, cdn.openai.com/better-language-models/cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

Rogers, Authors:Anna. "A Primer in BERTology: What we know about how BERT works" 9 Nov. 2020, arxiv.org/abs/2002.12327.

"Top 5 AI Detection APIs" *Sapling*, sapling.ai/ai-detection-apis.

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

Appendix

- Python3 used along with matplotlib, requests for GUI
- Node.js used along with React for web-app
- EmailRep API used for email reputation web scraping provider
- Originality.ai API used for AI detection from plaintext and URL