Final Report for Syslab
TextVerify - A Better AI Detector

Dev Bhatia

5/27/2025

Yilmaz - Period 3

Table of Contents

Abstract

This study is in the AI Detection field. It aims to create a better AI Detector that will be used to detect AI in students' writing. The topic of AI Detection is very important because it is crucial to make sure that students are submitting their own work and upholding integrity in the learning environment. Also, current AI Detectors don't work very well, so there is room for improvement in this field. The gap in current research involves the AI detection accuracy, with many detectors not being accurate in detecting AI. Some of these websites even have disclaimers that warn against using the AI Detection as an end all be all, as they aren't reliable. Our research aim was to make an AI Detector that would reliably detect AI content. The main method to do this involved Retrieval Methods, Doc2Vec encoding, and Cosine Similarity Scores.

I. Introduction

The main problem that was solved in this study was detecting AI. In today's day, AI is used very prominently across many fields and by many people across the world, specifically AI is used in the school setting by students. Many times, when doing writing assignments, students will put their question into an AI model and submit the result from the AI model. Also, some students have realized that paraphrasing AI generated text leads to AI Detectors not catching that text, so students have started to take AI content and then paraphrase it, then submit that so that the AI Detectors don't catch it. This problem leads to extreme difficulty for teachers to tell original work apart from AI, as current AI Detectors don't work very well [1]. Whenever teachers do use AI Detectors, they can give incorrect classifications on text, and that might cause a student to be accused of cheating even when their work was human written.

Another aspect of the AI problem is that students who use AI have a lack of overall knowledge in course content. They aren't learning the critical concepts that they should be when they use AI. The current circumstances around AI Detectors demonstrate that a better detection mechanism is needed to tell AI written text apart from human written text, and that is what this study attempted to do. We came up with this problem while thinking about problems that could be solved related to school and technology.

AI Detection is a big topic these days as AI is being used everywhere and is very popular. Also, AI is a topic that interests us as we are fascinated by how it works, so that added to our intrigue in doing this study. Lastly, we felt like making an AI Detector would help restore integrity in the school system, so that's why we decided to implement this as part of the project. The problem of AI Detection is fascinating because it is a problem that will define the next era of technology and school. AI is a relatively new concept, and it will continue to be very prominent in the future. The problem of AI and detection won't go away any time soon, so it's very important to find a solution to the problem for now and future generations. Also, it is something that is important to almost everyone. Every person that goes to school either uses AI or has heard of it, so it has relevance in schools today.

The AI Detection model I am proposing is TextVerify. The novelty in TextVerify is the way it uses Retrieval Methods, which is something that isn't used on current AI Detectors [4]. Retrieval Methods are a technique used to fetch relevant information from a large database. TextVerify uses a database of AI generations to tell whether the input text comes from AI. Along the way, TextVerify uses cosine similarity scores, finds matches to AI generations within the database, and then reports the result to the website. Also, it had a higher accuracy than any of the other most commonly used AI Detection models.

The main benefit of TextVerify is the way that it can successfully detect AI written text, this will allow teachers in school to reliably tell whether text has come from AI or not, and then hand out the fair punishment to that student. Whereas students that don't use AI won't be penalized like they would have with current AI Detectors that are used now. Later on, this will incentivize students to only submit work that is their own and that will create a more healthy learning environment where everyone is accessed based solely on their knowledge.

II. Background

Some of the existing AI Detectors are GPTZero, Copyleaks, and Crossplag.

The way GPTZero works is by looking at sentence structure, vocabulary, and different linguistic features in order to tell if text was AI generated or not. It also has deep learning models that are trained on large datasets of human and AI written text to classify text [12]. More generally, GPTZero looks at the patterns of the text to tell whether the text was AI generated or not. The amount of predictability in the text is a factor when deciding whether text was AI generated or not. The more predictable and less variable the text is, the more likely it was written by AI because AI writing has patterns in it, so GPTZero looks for patterns in the way the text is written [12].

Copyleaks checks for AI by looking at different patterns or irregularities in the text, Copyleaks compares the input text to human written text, and analyzes the similarities and differences to judge whether it was AI generated or not. It looks at the writing patterns of the text and compares that to human written text writing patterns, it does this going sentence by sentence through the input text [13].

Lastly, CrossPlag uses advanced Natural Language Process algorithms, and compares the input text with different human and AI written texts. At the end, the website outputs a percentage chance of the text being AI written or not [14]. The pro of the current AI Detectors is that they work well on older GPT generations [2]. The cons are that they fail on GPT 4, paraphrased AI generations, and overall they aren't very accurate [3].

III. Applications:

The main use of TextVerify is in education. Specifically, it can be used in writing assignments to check whether a student's text was written by AI. The teacher will input the text written by the student and the question the student was answering, and TextVerify will tell the teacher whether the text was written by AI or human written. It will be a big boost to school teachers, as it will be the first reliable source of AI Detection. Other AI Detectors aren't very reliable and cannot be solely used to tell if text was written by AI or not [3]. It can also be used by college professors and anyone else within an academic sphere for reliable AI Detection.

IV. Methods:

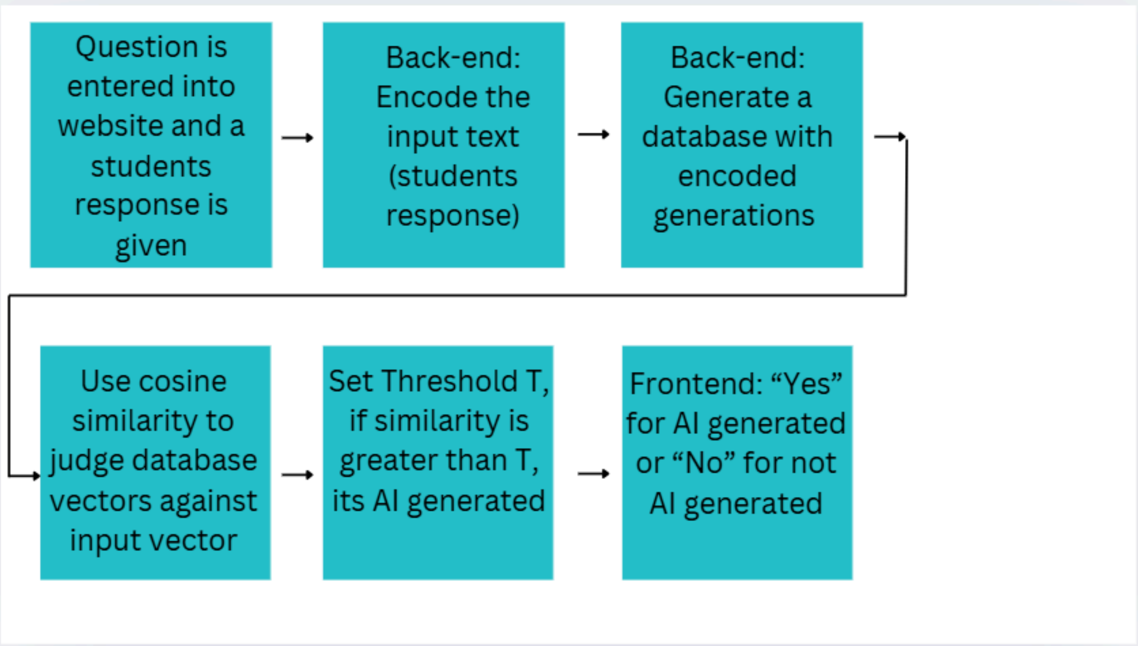

Question is entered into website and a students response is given
Back-end: Encode the input text (students response)
Back-end: Generate a database with encoded generations
Use cosine similarity to judge database vectors against input vector
Set Threshold T, if similarity is greater than T, its AI generated
Frontend: "Yes" for AI generated or "No" for not AI generated

Figure 1: Systems Diagram

High Level Overview:

As seen in Figure 1, the input to TextVerify is a question and a student response to that question. The question is the prompt for the specific writing task, and the student's response is being checked for being AI written. At the end, the TextVerify will output whether the text was AI generated or not.

After taking in the inputs, the next step in TextVerify is encoding the input text using Doc2Vec. This means taking the document and making a vector that represents the semantics of the document [6]. This happens through the usage of a Neural Network. Then, TextVerify generates a database of AI generated pieces of text, and then encodes those pieces of text with Doc2Vec. Later, Cosine Similarity score is used to compare the vectors in the database with the input text vector, and if the similarity between the input text vector and any database vectors is above a threshold, then the website outputs “AI Generated”. If the highest similarity is lower than the threshold, then the output on the website is “Not AI Generated”. This process is outlined in Figure 1 above.

Doc2Vec explained:

Doc2Vec is the encoding scheme that was used in this project to learn a fixed length vector representation from a document [6]. It was used to make the vectors that will go in the database, as well as to learn a vector for the input text document. Each document is eventually assigned a unique document vector. Doc2Vec works through a Neural Network. Data comes into the Neural Network in the form of documents, these documents are given random document vectors that update while training, eventually representing the semantics of the document [6]. Generally, Doc2Vec gets a window of words in a document, the input is context words along with a document vector, and the goal is to guess the middle word of the text window through the Neural Network [6].

Doc2Vec goes over each document in the set of documents (the input text document, as well as the documents that were generated to be part of the database), and initially it sets the Document Vector to random. Then Doc2Vec goes over each window of words in that document. A window of words is a length 10 string in the document, and there are a few of these within a given document. Doc2Vec has a goal of guessing the middle word in the window of words given the other 9 words and a document vector [6]. For that specific window of words, the input to the neural network is the initially random document vector and word vectors representing the 9 words in the text window that do not include the middle word [6]. Those word vectors are set to random. For example in the text window, “The American colonies declared independence from British rule in 1776” the middle word is “from”, and the other 9 words in that window represent inputs that are initially set to random vectors. Then those 9 word vectors along with the document vector are averaged together, and the ReLU function is then applied. Finally, a probability distribution is outputted by the neural network, this is then compared to the actual probability distribution. The actual probability distribution is an array over the entire vocabulary, with zeros in every spot, except a 1 in the spot where the middle word is. The actual probability distribution is compared with the probability distribution that the Neural Network produced, and then error propagation and gradient descent is done in order to minimize the error to the actual probability distribution [6]. This is done for every window of words in a document, and for every document that is in the set of documents.

Eventually, after multiple iterations, Doc2Vec adjusts the document vector and the word vectors to minimize the output prediction error. This means that the document vector will align with the semantics of the actual document. As multiple documents are trained with Doc2Vec, documents with similar ideas and words will be grouped together in the vector space for the purpose of later comparison [6].

Database:

The database is generated by pulling AI Generated text from ChatGPT 4o mini’s API on the question that was entered in by the teacher. The teacher enters in a question that the input text was written on and then that question is sent to ChatGPT 4o mini’s API, and 20 responses are generated. Those responses are then sent back to TextVerify, where they are encoded using Doc2Vec and then inserted into the database.

Cosine Similarity:

After the database has been generated and the input text is encoded with Doc2Vec, then Cosine Similarity is used. The formula for Cosine Similarity is seen in Figure 2. Cosine Similarity is a mathematical equation that is used to compare how alike two vectors are [7]. The formula gives a value between 0 and 1, where a similarity near 0 means the vectors are not very similar and a similarity near 1 means the vectors are similar [7]. All the Doc2Vec encoded vectors in the database are compared to the Doc2Vec representation of the input text, and if the highest cosine similarity score between a given database vector and the input vector is higher than the Threshold (T), then the input text is considered AI generated and that is outputted on the website.

$$\cos\theta = \frac{\vec{a}\cdot\vec{b}}{\|\vec{a}\|\|\vec{b}\|}$$

$$\|\vec{a}\| = \sqrt{a_1^2 + a_2^2 + a_3^2 + \cdots + a_n^2}$$

$$\|\vec{b}\| = \sqrt{b_1^2 + b_2^2 + b_3^2 + \cdots + b_n^2}$$

Figure 2: Cosine Similarity

Pseudocode and Doc2Vec diagram:

- Go over each document in the set of documents
  - Initialize the Document Vector to random
  - Go over each window of words (length = 10 words) in the document
    - Example window of words that goes into the Neural Network
      - “The American colonies declared independence from British rule in 1776.”
    - Input
      - Document Vector
      - Rest of the context words
        - “The American colonies declared independence British rule in 1776” not including middle word “from”
        - Each word is translated into random word vectors
    - Concatenation/average
      - Average the word vectors and Document Vector (then apply ReLU)
      - Output is a probability distribution over the vocabulary, the word with the highest probability is chosen as the neural network’s predicted word.
      - This is then compared with the middle word.
    - Do backpropagation to minimize the error to the actual word

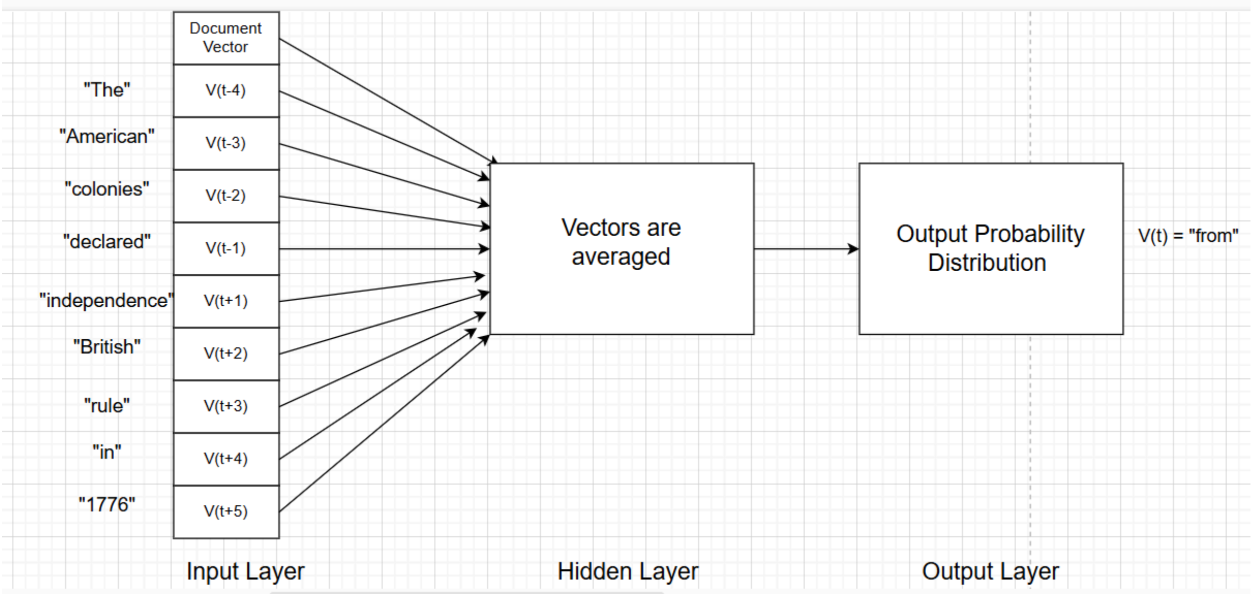


Figure 3: Doc2Vec Neural Network diagram

Explanation of Doc2Vec structure:

Doc2Vec is a 2 layer neural network. It takes in 10 inputs: document vector and 9 word vectors. Each is initialized to random and is updated with training [6]. The final output of the neural network is a probability distribution. A probability distribution is a vector with probabilities for each word in the dictionary, which is compared to the actual probability distribution and an error is calculated. Then, backpropagation and gradient descent are used to minimize the error [6]. This process is shown in Figure 3.

Reasoning for methods:

TextVerify uses these specific methods because they are novel and in initial research, they seemed to be really good pieces of defense against AI Generated text [4]. The initial research done shows that retrieval methods are a reliable way to check against AI content [4]. We took inspiration from the approach in this article [4], but added in Doc2Vec encoding and usage of Cosine Similarity scores along with a Threshold.

We decided to use Doc2Vec because it was one of the best ways to encode text to vector representations. It has proven results as a good Detection Mechanism. We decided on using Cosine Similarity as it is one of the best ways to compare how similar two vectors are [7], and that's what is needed after Doc2Vec encoding has been applied.

Finally, a Threshold was used because there needs to be a bar for detection of AI. A Cosine Similarity Score above the Threshold means its AI Generated, and a Cosine Similarity Score below that Threshold means its not AI generated, this is checked for every vector in the database.

We previously tried to use the Word2Vec encoding strategy, but that didn’t work because it only encoded singular words to vectors, and not entire documents. We spent a decent amount of time researching Word2Vec encoding techniques, but weren’t able to implement them because they specifically encode words and not documents as a whole.

We also tried integrating Chat GPT’s API into TextVerify multiple times until it worked. It was a tedious process as many online tutorials didn’t work, but after many tries we were able to get it to work. We got the error of GPT being outdated, but that was solved after we upgraded to the newest version of GPT.

Lastly, we experimented with using Contrastive Learning along with Retrieval Methods, but ultimately decided not to use them because they complicated the approach, and didn’t necessarily offer better results.

Given more time, we could have added other detection methods like watermarking and statistical outlier methods, these could have improved text detection even further. Also, we could have added expanded text detection to other areas besides the academic sphere, and added detection in other languages. This would have increased the broad functionality of the TextVerify and helped it be used in different spheres. Given a chance to totally redo this project, we would have addressed coding detection as well as text detection, that is also an important educational issue. Many students use AI to get coding assignments done, so that could have been addressed if I were to completely redo this project.

V. **How to run**:

In order to run TextVerify, use the Github repository for the project [5]. The first thing one has to do is create a Chat GPT API Key, and then export that as an environment variable on the terminal [8]. Then, one has to install the SDK using pip, this is done by typing in “pip install openai”. After this, run the commands “python reset_db.py” and “python app.py” in order to start the program on a local web server. Lastly, go to TextVerify’s website [15], and follow the instructions on the website to proceed.

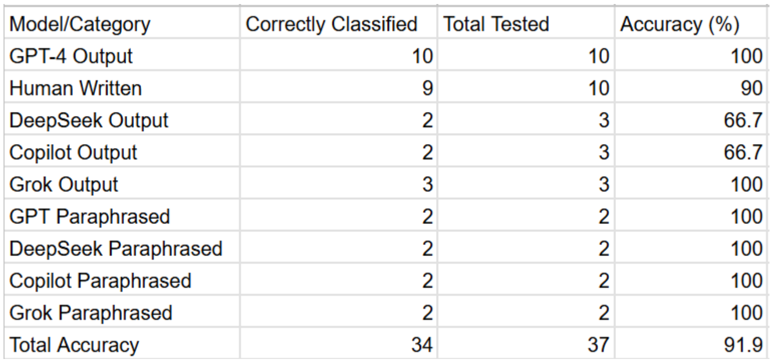
VI. **Results**:

TextVerify was tested on multiple pieces of text from different AI's including GPT-4, DeepSeek, Copilot, and Grok. It was also tested with paraphrased AI written content, and human written text content.

Overall, TextVerify got a 91.9% accuracy in detecting text generated by AI, as shown by Figure 4. Specifically, TextVerify was tested on 10 pieces of text that were GPT-4 generated, 9 pieces of text that were generated by other AIs,10 pieces of human written text, and 8 pieces of paraphrased AI text. It performed very well in all of those categories.

TextVerify got 100% accuracy in GPT-4 generated text, 90% accuracy in human written text, 78% accuracy in text written by other AIs, and 100% accuracy in detecting paraphrased AI content. In all, this resulted in a 91.9% accuracy. TextVerify takes in text that could have been directly taken from an AI text generator or paraphrased from an AI text generator and TextVerify tells the user whether the input text was AI generated or not.

TextVerify got better performance than current AI Detectors, as it was more reliable and got a much improved accuracy score. The picture below shows the results of testing TextVerify on different types of text and the results.



Model/Category	Correctly Classified	Total Tested	Accuracy (%)
GPT-4 Output	10	10	100
Human Written	9	10	90
DeepSeek Output	2	3	66.7
Copilot Output	2	3	66.7
Grok Output	3	3	100
GPT Paraphrased	2	2	100
DeepSeek Paraphrased	2	2	100
Copilot Paraphrased	2	2	100
Grok Paraphrased	2	2	100
Total Accuracy	34	37	91.9

Figure 4: Results

VII. **Limitations**:

Some areas not addressed in this study include text not generated by students, text with a shorter number of words, not using the watermarking and statistical outlier methods, and text written in other languages. Specifically, TextVerify only considered text that was generated for academic purposes, it didn't look at text that was used for any other purpose. Also, TextVerify didn't work on text that was of a shorter number of words. Shorter number of words means less than 50. Additionally, TextVerify didn't use statistical outlier and watermarking methods. These methods are used on other commonly used AI Detectors. Lastly, TextVerify only did AI Detection in English, it didn't detect AI in any other language.

VIII. **Conclusion**:

We solved the problem of AI Detection using Retrieval Methods. The key takeaway from this study was that We were able to establish a reliable AI detection mechanism that was able to consistently detect AI content. The main findings include a robust accuracy in detecting AI written output and paraphrased output, while maintaining that human written text was indeed human written. In total, the accuracy was 91.9%.

This research has the potential to be a breakthrough for educators in the educational sphere, they will be able to use TextVerify as a way to maintain integrity in their classrooms and help assess students only on their knowledge.

Some limitations of TextVerify include text not generated by students, not using watermarking and statistical outlier methods, text with a shorter number of words, and text written in different languages.

Future work to solve these limitations includes using retrieval methods for different languages and in different areas besides academics, as well as using retrieval methods along with statistical outlier methods and watermarking to detect shorter pieces of text.

IX. **Future Work**:

Areas of future research for this project include AI generated text not used in academics, using other AI detection methods along with Retrieval Methods for shorter text classification, and using Retrieval Methods in other languages.

The first area of future work that can be done related to this project is detecting AI written text in other areas besides academics. With TextVerify, the main focus of the study was detecting AI in academic activities, such as writing essays. This novel approach can be applied to many different fields besides that, such as the workforce and different jobs. It should be very helpful in those areas as well.

TextVerify can also be used in any realm where a person wants to test whether text was written by AI or not. A second future piece of work is using Retrieval Methods along with watermarking and statistical outlier methods in order to make shorter text detection better. Shorter text means 50 words or less. This means that a future researcher can take TextVerify and add watermarking and statistical outlier methods to their AI judgement process, and that should improve the detection mechanism for shorter pieces of text.

Lastly, Retrieval Methods can be used in detecting AI for other languages besides English. This project only covered English, but AI generation can be a problem in different languages as well, so future research can experiment with Retrieval Methods in languages such as French, Spanish, and German to reliably detect AI in those languages.

References:

[1]A. M. Elkhatat, K. Elsaid, and S. Al-Meer, “Evaluating the efficacy of AI content detection tools in differentiating between human and AI-generated text,” *International journal for educational integrity*, vol. 19, no. 1, Sep. 2023, doi: https://doi.org/10.1007/s40979-023-00140-5

[2]A. Singh, “A Comparison Study on AI Language Detector,” *IEEE Xplore*, Mar. 2023, doi: https://doi.org/10.1109/ccwc57344.2023.10099219

[3]M. Perkins *et al.*, “Simple techniques to bypass GenAI text detectors: implications for inclusive education,” *International Journal of Educational Technology in Higher Education*, vol. 21, no. 1, Sep. 2024, doi: https://doi.org/10.1186/s41239-024-00487-w

[4]K. Krishna, Y. Song, M. Karpinska, J. Wieting, and M. Iyyer, “Paraphrasing evades detectors of AI-generated text, but retrieval is an effective defense Mohit Iyyer,” Oct. 2023, doi: https://doi.org/10.48550/arXiv.2303.13408

[5]https://github.com/devsb123/AI-Detector-using-Retrieval-Methods

[6]G. Shperber, “A gentle introduction to Doc2Vec,” *Medium*, Nov. 05, 2019. Available: https://medium.com/wisio/a-gentle-introduction-to-doc2vec-db3e8c0cce5e

[7]A. Prakash, “Understanding Cosine Similarity: A key concept in data science,” *Medium*, Sep. 21, 2023. Available: https://medium.com/@arjunprakash027/understanding-cosine-similarity-a-key-concept-in-data-science-72a0fcc57599

[8]“OpenAI Platform,” *Openai.com*, 2025. Available: https://platform.openai.com/docs/quickstart?api-mode=chat

[9]E. Tian, “GPTZero,” *gptzero.me*, 2022. Available: https://gptzero.me/

[10]“Copyleaks: AI & Machine Learning Powered Plagiarism Checker,” *copyleaks.com*. Available: https://copyleaks.com/

[11]“Crossplag,” *app.crossplag.com*. Available: https://app.crossplag.com/individual/detector

[12]V. Chen, “How Do AI Detectors Work? | GPTZero,” *AI Detection Resources | GPTZero*, Oct. 14, 2024. Available: https://gptzero.me/news/how-ai-detectors-work/

[13]“AI Content Detector FAQs How It Works Understanding the Results Detection Capabilities & Limitations.” Available: https://copyleaks.com/wp-content/uploads/2023/05/ai-content-detector-faqs.pdf

[14]Agnesa Nuha, “Detecting if a text is AI generated - Crossplag,” *Crossplag*, Dec. 19, 2022. Available: https://crossplag.com/detecting-if-a-text-is-ai-generated/. [Accessed: Apr. 24, 2025]

[15]http://127.0.0.1/