Plagiarism Checker Based on Machine Learning and OpenAI

Chaitra Boggaram
Graduate Engineering
Santa Clata University
California, USA
cboggaram@scu.edu

Oluwabusayo Omotosho
Graduate Engineering
Santa Clara University
California, USA
oomotosho@scu.edu

Tenghjiao ZHU
Graduate Engineering
Santa Clara University
California, USA
tzhu2@scu.edu

***Abstract* - Plagiarism has become a significant concern in academic writing and publishing, leading to the development of various plagiarism detection systems. However, most of these systems utilize specific algorithms to construct their plagiarism detection mechanisms. In this paper, we present a plagiarism checker specifically designed for essays generated using ChatGPT, an advanced language model developed by OpenAI using the PyTorch machine-learning library written in Python. Our experiments demonstrate that our plagiarism detection system outperforms some existing systems in detecting instances of plagiarism. By leveraging the capabilities of ChatGPT, we have developed a unique and effective tool for maintaining academic integrity and preventing plagiarism in writing.**

***Keywords* - plagiarism checker, machine learning, OpenAI, ChatGPT.**

I. INTRODUCTION

The use of plagiarism detection software is becoming more and more crucial. A plagiarism checker is a technology that compares a supplied work to a database of sources to look for any possible instances of plagiarism.

There are several methodologies to build a plagiarism checker including string matching, a bag of words, machine learning, or hybrid. Each method has its strengths and weaknesses [1-3]. String matching is the most basic but not good at detecting paraphrasing. A bag of words can detect both copying and paraphrasing but has a high rate of false positives. Machine learning can detect plagiarism well but requires lots of training data. Hybrids can improve accuracy but can be complicated.

Several studies offer fresh approaches to implementing various sorts of anti-plagiarism systems [4 - 11] based on these common strategies. Details about each method will be explained in the following sections.

II. RELATED WORKS

A. Existing Methodology

Many articles have been written about the mechanism of plagiarism, which has created a solid foundation for the development of anti-plagiarism technologies.

The most common and harmful types of plagiarism around the world, particularly in Ukraine, were outlined by Shkodkina and Pakauskas[12]. The four subsets of their criterion were affordability, material (format) support, functionality, and showcasing. However, nearly all criteria approaches and scoring were binary. Also, for both text-based and source code comparability findings, Sobhagyawati and Jharotia introduced more than 30 and 10 frameworks [12], respectively. These investigations aimed to provide a list, illustration, and coverage of the most popular devices available rather than a correlation between said frameworks.

Using categorization and computational complexity, Lancaster [17] computes similarities. Statistical metrics can be language-independent or language-sensitive, according to Gruner's work [18]. Moreover, the research on the ability to distinguish between four different types of obfuscation, including back-translation, basic copy-paste, random obfuscation, and summarization is proposed by Vani and Gupta [12]. Their studies show that the information on eight systems and their main characteristics is accurate across systems when no obfuscation is employed.

Hector Garcia-Molina proposed a method called the Stanford Copy Analysis Method (SCAM), which is efficient in recognizing different kinds of plagiarism [19]. Also, Chow and Rahman [20] devised another method for plagiarism detection by using a tree-structured representation and an ML-SOM (Machine Learning Self-Organizing Map). Moreover, M. Zini suggested an approach to locating word or phrase clusters in different by using specific words or keywords [21].

Elhadi and Al-Tobi proposed a duplicate detection method that transformed each document into a set of parse trees that represent its grammatical structure [22]. Furthermore, Grozea and Popescu [23] devised a technique that is intended to be language-independent and works with both single-language and multilingual materials for determining how closely two documents resemble one another. Gelbuk used data from dictionaries or thesauri to produce nodes that represent concepts or meanings and edges that show how those concepts are related to one another [24].

These techniques lay the groundwork for many different types of plagiarism detection systems.

B. Existing Detection System

Based on these methodologies mentioned above, there are several tools [12] built by universities and institutes available to detect plagiarism,

Compilatio is plagiarism detection software that compares submitted documents against a large database of sources, both online and offline. To detect plagiarism, the software compares language, sentence structure, and even formatting. Also, Copyleaks makes it simple to access a wide range of content from various genres, making plagiarism a common occurrence. This can be concerning for authors, whether professionals or academics and emphasizes the importance of understanding how anti-plagiarism software works.

DupliChecker.com compares input text to a massive database of billions of web pages on the internet. It meticulously examines your document sentence by sentence to ensure complete accuracy, leaving no room for any potential matches to be overlooked [12]. Furthermore, Google Originality Reports can be used to verify the authenticity of Google Docs or Slides by comparing them to online web pages and books. Any detected sources are linked within the report, and any uncited text is highlighted for review.

PlagiarismCheck.org employs cutting-edge AI techniques designed to detect and track AI-generated content that makes it difficult to distinguish it from the human-generated text. PlagiarismCheck.org can provide highly accurate results and avoid false positives by analyzing a variety of parameters and constantly updating our algorithms. Moreover, GPTZero is a classification model that predicts whether a document was written by an AI model or by a human, providing predictions on a sentence, paragraph, and document level. This model is written by Edward Tian, a computer science student from Princeton University, and is said to be "quick and efficient" to decipher whether a human or ChatGPT authored an essay. GPTZero was created using Python.

These systems, despite their widespread use, have several shortcomings, such as lengthy wait times and high mistake rates. To avoid these negative effects, a novel strategy must be put forth.

III. Evolution of ChatGPT

The last few years have seen significant development for ChatGPT. It has been considered the most sophisticated chatbot that has been made [13 - 14] because this chatbot can manage various tasks, for instance, generating code snippets, composing papers, and stories, and performing complex mathematical calculations. The ability to acclimate to any request without getting ready has made it a champion among other language models.

ChatGPT was trained on humongous datasets which utilized deep neural networks to frame profound learning brain networks demonstrated after the human brain. This permits ChatGPT to learn patterns and connections in the message information to anticipate what text comes next in some random sentence. However, it does not operate at the sentence level; instead, it creates text that suggests possible words, sentences, and even paragraphs.

Figure 1 shows that retrieving a prompt from the database, exhibiting appropriate behaviors, and fine-tuning GPT-3 are the three steps in training a supervised model.

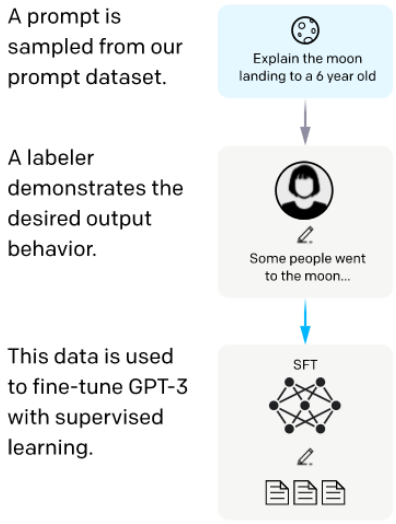

Figure 1 - Collect demonstration data and train a supervised policy. (OpenAI, 2019)

Figure 2 shows the reward model training process. Sample outputs from other models are in the initial stage. These outputs will then be manually sorted. After that train a rewarding model by using these sorted data.

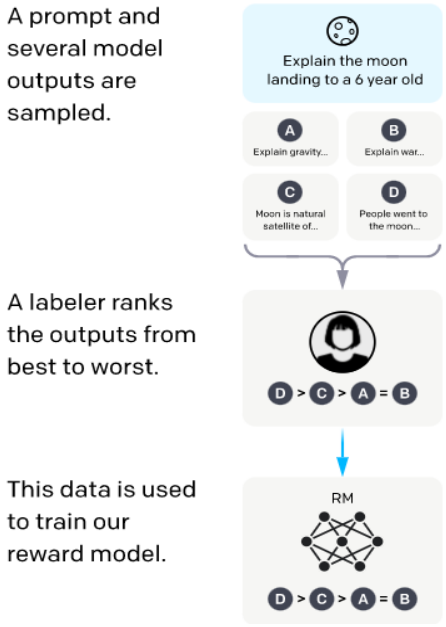

Figure 2 - Collecting Comparison data and training a reward model (OpenAI, 2019).

Furthermore, Figure 3 illustrates how to update a reward policy. Extracting a prompt from a database is followed by getting a new output from the existing reward model. Then, the reward model calculates a new reward for output, which is used to update the existing reward policy [15].

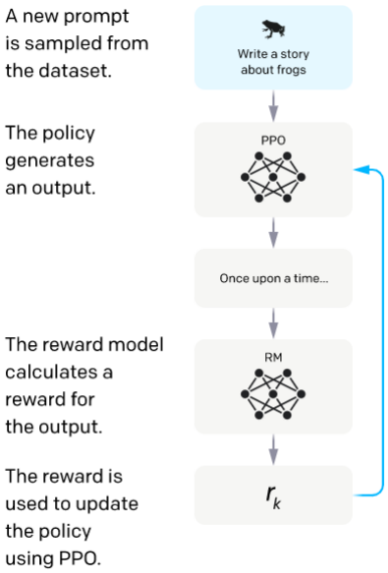

Figure 3 - Optimize a policy against the reward model using reinforcement learning (OpenAI, 2019)

ChatGPT has the potential to significantly enhance scholarly research and librarianship in unexpected and ground-breaking ways. Therefore, it is important to think about how to employ this innovation conscientiously and morally and to discover ways to cooperate with it.

IV. Model Proposed

Our checker will feature a user-friendly interface made with Flask, HTML, CSS, and Python where users can easily submit the file to check for plagiarism with precise plagiarism detection techniques like text comparison and string matching. Moreover, text preprocessing and representation are done by using Python for removing the special characters, numbers, punctuations, and case conversion to keep only meaningful words that will be used as input for the plagiarism checker.

A. Methodology

For building the plagiarism checker, we use a website where we can upload the file to which plagiarism must be checked using Python as the programming language and HTML, and CSS for creating the web pages. Here is the structure for our checker.

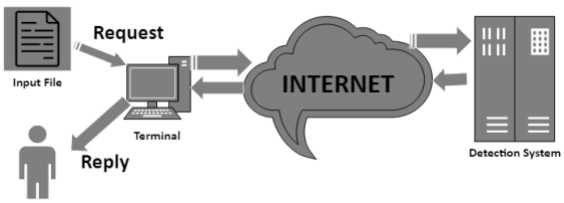

Figure 4 - Structure

Figure 4 illustrates the integration of plagiarism-checking requirements such as text preprocessing and representation. The uploaded text undergoes preprocessing by removing special characters, numbers, and punctuation, and converting the text to lowercase to retain only meaningful words. The output is then passed to the plagiarism checker to compute the similarities between the uploaded text and the text generated by ChatGPT using the cosine similarity algorithm. The result is displayed on the screen in a graphical format, indicating the similarity score between the texts. Alternatively, the user can download the findings in a file, which highlights the copied text and provides a similarity score.

B. Pseudo Code

1. Begin

2. Prompt the user to upload a file to the webpage

3. Check if the file was successfully uploaded

4. If the file was not successfully uploaded, terminate the program

5. Else, open the uploaded file in read mode

6. Read the contents of the file into a buffer

7. Close the file

8. Call the OpenAI API to process the buffer

9. Check if the API call was successful

10. If the API call was not successful, terminate the program

11. Else, receive the return value from the API call

12. Create a new file called "output.txt"

13. Check if the file was successfully created

14. If the file was not successfully created, terminate the program

15. Else, open the output file in write mode

16. Write the return value to the output file

17. Close the output file

18. Prompt the user to download the output file

19. End

C. Subsystems

We have proposed a comprehensive structure for our plagiarism checker, which includes four key systems: an I/O system, an interactive system, a computation system, and downloading system. Each system is designed to perform specific tasks and functions to ensure the accurate and efficient analysis of documents. Here are the details of each system.

1) I/O System

This structure has the responsibility of managing document input and output. The user login, file upload and verification, and sending the validated file to the interactive system for additional processing are the three main parts of the I/O system.

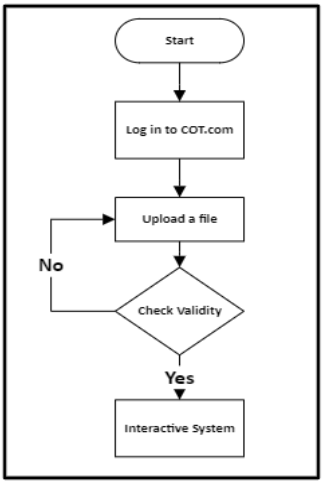

Figure 5 - I/O system

Figure 5 demonstrates that the I/O system is composed of three key components. Users can upload the file or copy-paste the text that they wish to check for plagiarism in our system. During the upload process, our checker will verify the file's extension to ensure compatibility (.txt or .docx) and file size (not more than 2000 characters). Once the file has been successfully uploaded and verified, it will be passed to the interactive system for further processing. This multi-step process ensures that only valid and compatible files are analyzed by our checker, leading to more accurate and reliable results.

2) Interactive System.

This system analyzes the documents and generates relevant data for further processing. The interactive system comprises five components: document validity checking, keyword extraction, communication with OPENAI, storing return values, and passing parameters to the computation system.

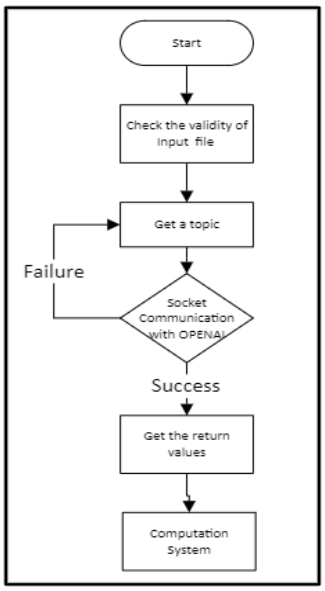

Figure 6 - Interactive system

Figure 6 illustrates that the interactive system comprises five main components. After the document has been validated, our checker will extract relevant keywords to help with the analysis. The communication component oversees enabling API contact between our checker and OPENAI so that data and information can be exchanged. The return numbers are kept in a static folder after the analysis is finished. The computation system uses these return values as parameters to analyze the data and produce the desired outcomes. This multi-step procedure makes sure that all pertinent data is correctly analyzed and processed, resulting in more thorough and trustworthy results.

3) Computation System

This system is responsible for processing the data generated by the interactive system and generating the final plagiarism analysis results. The computation system processes the parameters received from the interactive system and generates a report highlighting any instances of plagiarism.

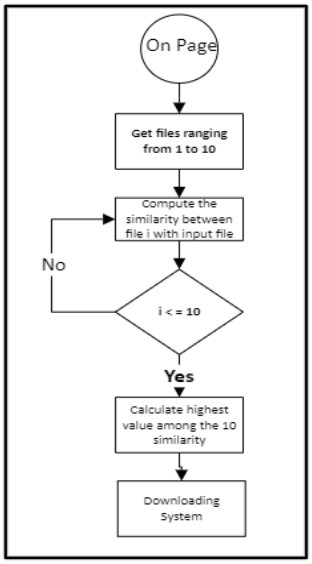

Figure 7 - Computation System

Figure 7 illustrates the components of the computation system, which work together to achieve this goal. The computation system first determines the similarity between the input file and the files generated using OPEN AI. The system uses the sklearn machine learning module to calculate the similarity between files. Once the similarity for each file has been determined, the system identifies the highest score among all the similarities. This highest score represents the targeted score, which will be used to pass as a parameter to the downloading system.

4) Downloading System

This system is responsible for providing users with access to the final plagiarism analysis report. Once the report is generated, the downloading system allows users to download the report in a user-friendly format for further analysis.

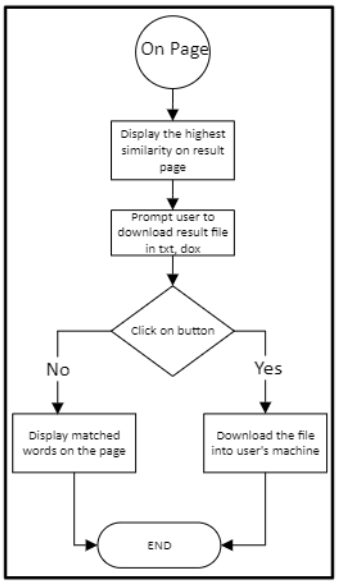

Figure 8 - Downloading system

Figure 8 illustrates that the computation system is divided into two main components. Firstly, the similarity between the input document and the sources identified as potential instances of plagiarism is displayed on the page in the form of a pie chart. This allows the user to visualize the results of the plagiarism analysis quickly and easily. Secondly, there is a download button available for users to download the plagiarism checking result generated by our checker. This allows users to access a detailed report outlining any instances of plagiarism detected in their document. The combination of these two components provides users with a comprehensive understanding of the plagiarism analysis results and allows for further analysis and evaluation.

V. Experiments

A. *Experiment with other Plagiarism checkers*

This paper uses ChatGPT to generate an essay with 2000 characters for this experiment. Subsequently, we subjected the generated essay to a plagiarism check using 10 different websites. The entire process is illustrated in Figure 9, which outlines the step-by-step process of generating the essay, uploading it to the plagiarism checker, and analyzing the results. This experiment allowed us to evaluate the efficacy of our plagiarism checker by comparing its results to those generated by other existing plagiarism checkers.

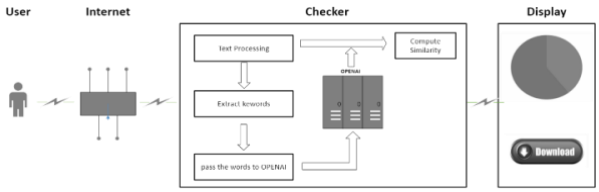

Figure 9 - Flow Chart

When comparing different plagiarism-checking tools, COT stood out as the most effective at identifying instances of plagiarized content in an essay. While Copyleaks was also able to identify some instances of paraphrasing, it did not catch as much plagiarism as COT. This suggests that COT's algorithms and machine learning techniques are more advanced and capable of detecting more nuanced forms of plagiarism.

In contrast, the other tools that were used were unable to detect any instances of plagiarism at all. This may indicate that these tools are not as reliable or accurate as COT and Copyleaks and may not be suitable for detecting more subtle forms of plagiarism. It's important to note, however, that no plagiarism checker is foolproof, and it's always recommended to use multiple tools to ensure the most comprehensive analysis of a document.

TABLE I

Comparing Plagiarism Checkers for Essays		
Websites	***Plagiarized Percentage***	***Detection Time (in Seconds)***
https://www.duplichecker.com/	0%	40.71
https://smallseotools.com/plagiarism-checker/	0%	50.22
https://www.quetext.com/plagiarism-checker	0%	42.51
https://www.plagiarismchecker.co/	0%	65.05
https://plagiarismdetector.net/	0%	37.66
https://copyleaks.com/plagiarism-checker	68.8%	77.33
https://www.editpad.org/tool/plagiarism-checker	0%	15.55
https://grammica.com/plagiarism-checker	0%	10.68
https://rewriteguru.com/plagiarism-checker/	0%	88.78
COT	**83%**	**100.90**

Table 1 illustrates the results of the plagiarism checkers and the time taken by them to compute the results. Out of 10 websites, only 2 were able to detect plagiarism.

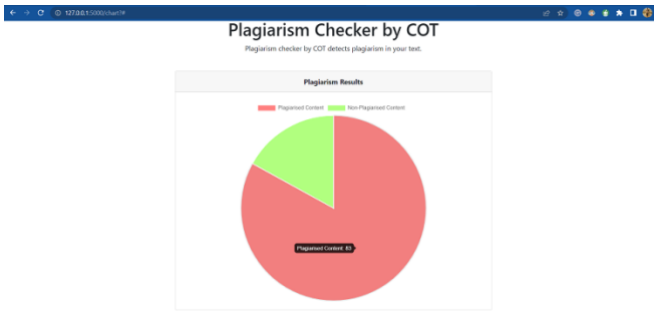

Figure 10 - COT Plagiarism results

Figure 10 illustrates how COT detected most of the plagiarism by following a multi-step process. Firstly, COT extracts the title from the input file, then queries ChatGPT for 10 different results. These results are analyzed and compared

to the input file to identify any similarities or matches between the content. By utilizing ChatGPT, COT can tap into a vast amount of language data and knowledge to provide more accurate and reliable plagiarism detection.

Figure 11 – Highlighted content

Figure 11 further illustrates how COT highlights the matching content in the input file. This feature helps users quickly identify any instances of plagiarism and review the content in question. By providing visual aids, COT makes it easier for users to understand and address any issues with their work.

Overall, COT's multi-step process and integration with ChatGPT make it a powerful tool for detecting plagiarism. The ability to extract titles and provide visual aids for matching content helps users quickly identify and address any potential issues, promoting academic and professional integrity.

B. *Experiment on GPTZero*

We conducted multiple experiments, but the results of these two experiments were particularly interesting.

1) GPTZero with ChatGPT generated content.

In this experiment, GPTZero was able to detect the entire content written by ChatGPT as illustrated in figure 12, but it failed to identify content that was modified or paraphrased to some extent. Specifically, when we made modifications to the content or used paraphrasing techniques, GPTZero was not able to identify it as having originated from an AI model as illustrated in figure 13.

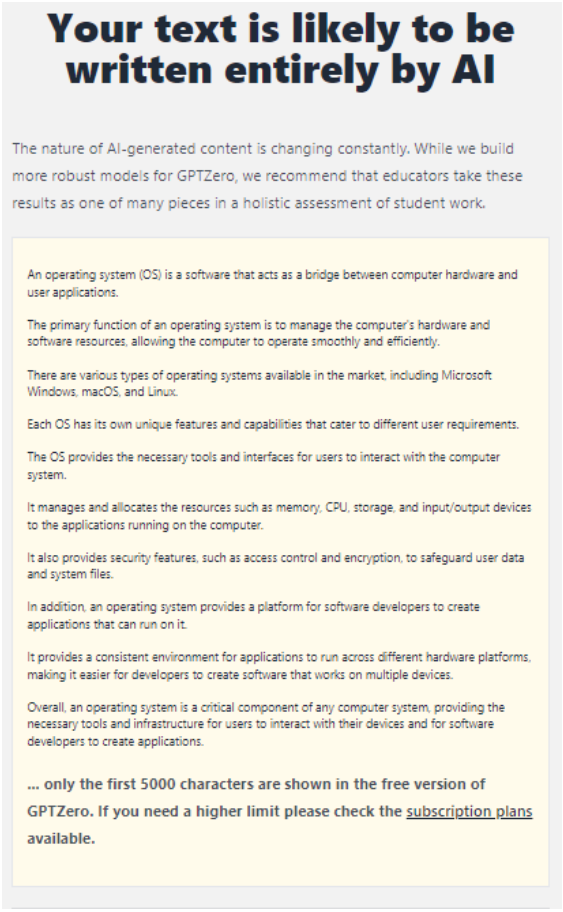

Figure 12 – Content detected written by ChatGPT

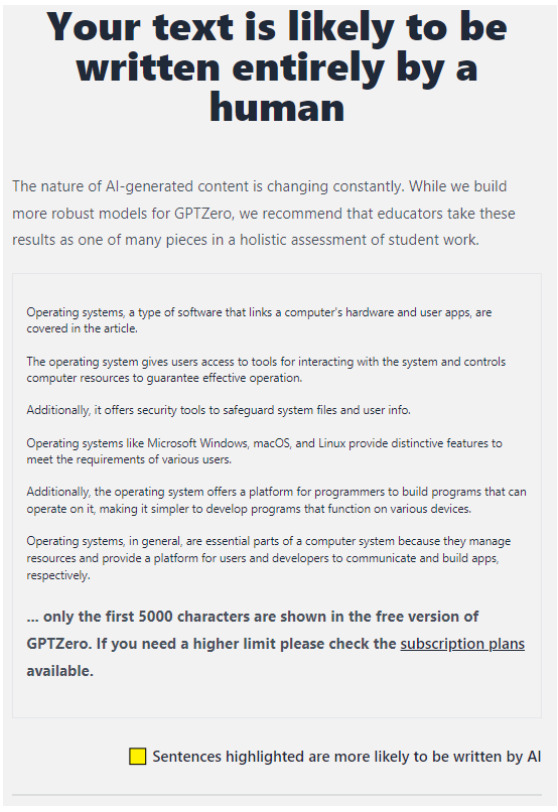

Figure 13 - Content not detected after modifying

2) GPTZero with Modern Operating Systems book

In this experiment, we found that GPTZero gives very mixed results when we tested the GPTZero by pasting the content from the book "Modern Operating Systems" fourth edition by Andrew S. Tanenbaum and Herbert Bos, which was written in 2014. GPTZero states that one-third of this given content is AI written, and this is only the first page taken from the textbook. The textbook which was written in 2014 is not AI written, and this makes it obvious that this is a false positive. Figure 14 illustrates the results of GPTZero generated for the first page of the Introduction chapter in the textbook.

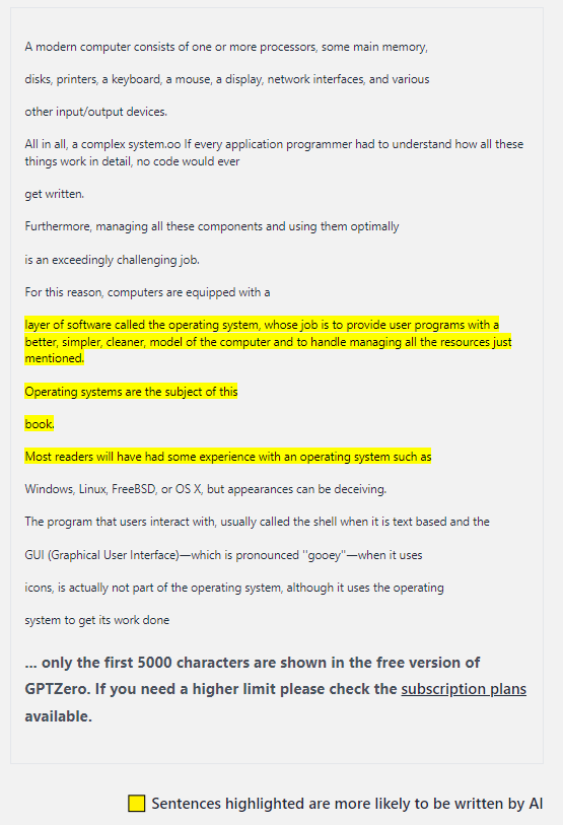

Figure 14 – Results from GPTZero

VI. Conclusion

The COT plagiarism checker will check for text similarities, string matches, and semantic analysis between them by importing the OPENAI module to Python, we will generate the results from ChatGPT which are similar to the uploaded file for comparison. Then we check for a similarity score using the cosine similarity between the input file and the ChatGPT result file. The result generated by the plagiarism checker will highlight the similarities between the uploaded file and the result file that we get from ChatGPT is provided in a graphical representation for better viewing, while the generated report is also available for the user to download afterward.

The experiment aims to evaluate the efficacy of plagiarism checkers and the accuracy of GPTZero which is a classification model that predicts whether a document was written by an AI model or by a human. In the experiment, we checked plagiarism on an essay generated by ChatGPT using 10 different plagiarism checkers and compared their results. The experiment found that COT and Copyleaks were the most effective at detecting plagiarism, while the other tools were unable to detect any instances of plagiarism. GPTZero has tested with ChatGPT-generated content and the book "Modern Operating Systems" fourth edition by Andrew S. Tanenbaum and Herbert Bos, and the experiment found that GPTZero was able to detect the entire content written by ChatGPT, but failed to identify content that was modified or paraphrased to some extent. In addition, GPTZero gave mixed results when tested with the textbook "Modern Operating Systems" fourth edition, giving a false positive.

However, COT gave some false positives where human-written content was flagged as content written by an AI model as human-written content has similar phrases to AI-generated content. To reduce false positives when detecting plagiarism between human-written content and AI-generated content, the threshold can be adjusted for the similarity score, where documents with a high level of similarity will be flagged as potentially plagiarized. We can also try to implement datasets in the COT plagiarism checker, by collecting a diverse set of documents, preprocessing them to remove unnecessary formatting, and split them into training and testing sets, applying similarity measures to those documents, such as n-gram analysis, syntactic similarity, and semantic similarity, to identify similarities between them. Finally, to evaluate the performance of COT using the testing set to identify areas for improvement.

VII. References

[1] Naik, Ramesh R., Maheshkumar B. Landge, and C. Namrata Mahender. "A review on plagiarism detection tools." International Journal of Computer Applications 125.11 (2015).

[2] G. EasonBruno, Andrea, et al. "Distributed anti-plagiarism checker for biomedical images based on sensor noise." New Trends in Image Analysis and Processing–ICIAP 2017: ICIAP International Workshops, WBICV, SSPandBE, 3AS, RGBD, NIVAR, IWBAAS, and MADiMa 2017, Catania, Italy, September 11-15, 2017, Revised Selected Papers 19. Springer International Publishing, 2017.

[3] Meo, Sultan A., and Muhammad Talha. "Turnitin: Is it a text matching or plagiarism detection tool?." Saudi journal of anaesthesia 13. Suppl 1 (2019): S48-https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6398291/.

[4] Razı, Salim. "Development of a rubric to assess academic writing incorporating plagiarism detectors." Sage Open 5.2 (2015): 2158244015590162.

[5] Halak, Basel, and Mohammed El-Hajjar. "Plagiarism detection and prevention techniques in engineering education." 2016 11th European Workshop on Microelectronics Education (EWME). IEEE, 2016.

[6] Eaton, Sarah Elaine, et al. "An institutional self-study of text-matching software in a Canadian graduate-level engineering program." Journal of Academic Ethics 18.3 (2020): 263-282.

[7] Nova, Muhamad, and Westi Utami. "EFL STUDENTS'PERCEPTION OF TURNITIN FOR DETECTING PLAGIARISM ON ACADEMIC WRITING." International Journal of Education 10.2 (2018): 141-148.

[8] Kunschak, Claudia. "Multiple uses of anti-plagiarism software." The Asian journal of applied linguistics 5.1 (2018): 60-69.

[9] Al Jarrah, Abeer, Izzat Alsmadi, and Zakariya Za'atreh. "Plagiarism Detection based on studying the correlation between Author, Title, and Content." International Conference on Information Communication System (CICS). 2011.

[10] Niklander, Tiina. "How to avoid plagiarism?." Proceedings of AMICT 2010-2011 Advances in Methods of Information and Communication Technology: 87.

[11] Potthast, Martin, et al. "An evaluation framework for plagiarism detection." Coling 2010: Posters. 2010.

[12] Ali, Asim M. El Tahir, Hussam M. Dahwa Abdulla, and Vaclav Snasel. "Overview and comparison of plagiarism detection tools." Dateso. 2011.

[13] Okonkwo, C.W., & Ade-Ibijola, A. Chatbots applications in education: A systematic review. Computers and Education: Artificial Intelligence, 2, 100033. (2021)

[14] OpenAI. (2019). Learning to Follow Natural Language Directions in Unknown Environments. Retrieved from https://openai.com/research/instruction-following.

[15] Zhou, Ce, et al. "A comprehensive survey on pretrained foundation models: A history from bert to ChatGPT." arXiv preprint arXiv:2302.09419 (2023).

[16] GPTZero. (n.d.). GPTZero, from https://app.gptzero.me/app/welcome

[17] Lukashenko, Romans, Vita Graudina, and Janis Grundspenkis. "Computer-based plagiarism detection methods and tools: an overview." Proceedings of the 2007 international conference on Computer systems and technologies. 2007.

[18] Gruner, S., S. Naven. Tool support for plagiarism detection in text documents. Proceedings of the 2005 ACM Symposium on Applied Computing. pp. 776 – 781, 2005.

[19] Shivakumar, Narayanan, and Hector Garcia-Molina. "SCAM: A copy detection mechanism for digital documents." DL. 1995.

[20] Chow, Tommy WS, and M. K. M. Rahman. "Multilayer SOM with tree-structured data for efficient document retrieval and plagiarism detection." IEEE Transactions on Neural Networks 20.9 (2009): 1385-1402.

[21] Zini, Manuel, et al. "Plagiarism detection through multilevel text comparison." 2006 Second International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution (AXMEDIS'06). IEEE, 2006.

[22] Elhadi, Mohamed, and Amjad Al-Tobi. "Use of text syntactical structures in detection of document duplicates." 2008 Third International Conference on Digital Information Management. IEEE, 2008.

[23] Grozea, Cristian, Christian Gehl, and Marius Popescu. "ENCOPLOT: Pairwise sequence matching in linear time applied to plagiarism detection." 3rd PAN Workshop. Uncovering Plagiarism, Authorship and Social Software Misuse. 2009.

[24] Torres, Sulema, and Alexander Gelbukh. "Comparing similarity measures for original WSD lesk algorithm." Research in Computing Science 43 (2009): 155-166.