A 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 is a serious issue in academic writing and publishing, and the use of plagiarism has become increasingly important. There are lots of systems. However, most of them are using specific algorithms to build their systems. In this paper, we are building a plagiarism checker for essays written using chatGPT. ChatGPT is an advanced language model developed by OpenAI which is developed using PyTorch (open-source machine learning library) written in Python. The experiments show that our detecting system is better than some systems.

Keywords— plagiarism checker, machine learning, ChatGPT.

# Introduction

The use of plagiarism checker software has become increasingly important. A plagiarism checker is a tool that checks a given document against a database of sources to identify any potential cases of plagiarism. There are several methodologies to build a plagiarism checker, and this report aims to explore these methodologies in detail, discussing their implementation, advantages, and disadvantages.

Most plagiarism detection systems use one of four methods: string matching, bag of words, machine learning, or hybrid. Each method has its strengths and weaknesses. 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. Hybrid can improve accuracy but can be complicated. Details about each method will be explained in the following sections.

The string-matching methodology is the most basic methodology used in plagiarism checkers. This methodology involves comparing the words in each document against a database of sources. If the words match, it indicates the presence of plagiarism. This methodology is implemented using algorithms such as Levenshtein Distance, Longest Common Subsequence (LCS), and Jaccard Similarity. This methodology is relatively easy to implement, and it can identify direct copies of text, while this methodology is not effective in detecting paraphrasing, and it can result in a high rate of false positives. Here are the steps to implement a plagiarism checker using the string-matching methodology.

# RELATED WORKS

## Current Methodology

There are lots of papers studying the mechanism of plagiarism. Maurer et al. (2006) work on practical models for contrasting instruments incorporate the treatment of even satisfied and designs, the ability to prohibit explicit sources, and the confirmation of the references. Likewise, they saw how really summarizing, interpretation, and direct literary theft were distinguished. They arrived at the resolution that counterfeiting recognition devices are inadequate against reword, interpretation, and cases in which the web-based presence of the first source material is compromised. Also, they battle with extraordinary characters and even text. However, they gave no data on their procedure or testing corpus. Ali et al. (2011), compared the ability to analyze several documents in parallel, the ability to verify sentence structure and synonyms, and the database breadth (online and offline databases, the Internet, and books, journals, periodicals, PDF...) of five programs were all compared.

All of the results from their comparison were quite close, and the scoring methodology was not made clear.Both their approach and corpora were not disclosed. Vani and Gupta (2016), examined three of those tools for their capacity to recognize four different levels and techniques of obfuscation: simple copy paste, random obfuscation, back-translation (by machine-translating text from English to Hindi, then back to English), and finally summarization. They reported on eight systems in terms of their key features. When no obfuscation is used, their findings reveal that the detection is accurate across systems, but when other approaches are used, it drastically declines, reaching a similarity score of 0% for all three tools.

In their study, Birki et al. (2016) compared four tools by looking at a variety of technical and functional factors in addition to detection accuracy. The availability of an API (Application Programming Interface) or plugins for learning management systems, deployment options, the range of source retrieval and the ability to include personal content, costs and licensing modalities, and finally the ability to be used in an organizational setting by talking about the available user roles and authentication modalities, are some criteria. This study is additionally intriguing because it includes several metrics that tailor it for usage in Croatian universities and academic institutions. It takes into account the capability of authenticating with "AAI@EduHr e-ID", the authentication and authorization standard in science and higher education. Nahas (2017) looked at 15 tools, some of which are old or aren't being used. There was no genuine comparison to separate the tools other than a division into open (free) and paid programs. The study was restricted to a brief explanation of each tool and recommendations for its users.

Prior to conducting a comparison study on three systems based on factors that were determined to be the most important for users in Ukrainian universities and by discussing the benefits and drawbacks of each, Shkodkina and Pakauskas (2017) provided an overview of the most prevalent and serious forms of plagiarism worldwide and in Ukraine in particular. Affordability, material (format) support, functionality, and showcasing were the four subsets of their criterion (the ability to establish the discussion between teachers and students). Even when a score would have been more descriptive, such as for the paraphrasing detection, which was given a "Yes" for all three systems, practically all criteria's approach and scoring were binary (Yes or No).

Sobhagyawati (2017) and Jharotia (2018), each introduced more than 30 and 10 frameworks individually, for both text based and source code comparability discovery. Their studies did not plan to offer a correlation between said frameworks, however basically list, depict and cover the most utilized or referred to devices available, the free too the business ones.

## Existing Detection System

Compilatio is a plagiarism detection software that uses advanced algorithms to compare submitted documents against a vast database of sources, including online and offline materials. The software checks for similarities in language, sentence structure, and even formatting to identify instances of plagiarism.

Copyleaks provides easy access to a vast array of content from various genres, making plagiarism a common occurrence. This can be a cause of concern for authors, be it professionals or academics, and highlights the importance of understanding the workings of anti-plagiarism software.

DupliChecker.com scans your text against an extensive database of billions of web pages available on the internet. It meticulously examines your document sentence by sentence to ensure complete accuracy, leaving no possibility of overlooking any potential matches. The tool is available for free, but if you require multiple scans per day, you must register to use it. By creating a registered profile, you can access the tool anytime you want, without incurring any charges.

Google Originality Reports can be utilized by both you and your students to verify the authenticity of their work. By running a report, a student's Google Docs or Slides file is compared against online webpages and books. Any detected sources are linked within the report and any uncited text is flagged for attention.

PlagiarismCheck.org uses cutting-edge AI techniques, our solution is designed to detect and track AI-generated content, even in cases where it may be difficult to distinguish from human-generated text. By analyzing a range of parameters and constantly updating our algorithms, we are able to provide highly accurate results and avoid false positives, where human-generated text is mistakenly flagged as AI-generated. Our AI text detector is always improving, ensuring that you have access to the latest and most effective methods for detecting instances of AI-generated content use.

# Evolution of ChatGPT

ChatGPT has evolved deeply in the past few years. It has been considered as the most surprising chatbot that has anytime been made [11]. Amazingly, this chatbot has the ability to manage various tasks, for instance, making code scraps, making papers, stories, works and regardless, performing complex mathematical errands. The ability to acclimate to any request without getting ready-made I champion among other language models.

Graphical user interface

Description automatically generated

Figure 1 - Collecting Data

As indicated by Rudolph, Endlessly tan [11], ChatGPT has been pre-prepared on more than 40 terabytes of text. In straightforward math, this is near 40 million books in a fuel design. The early adaptations of GPT, GPT-1 and GPT-2 delivered in 2018 had 117 million and 1.2 billion boundary separately. GPT itself had 175 billion boundaries, the improvement was multiple times how much information that GPT-1 was prepared on which made GPT a beast and without equal. representing progressed Regular Language Handling (NLP) and controlled by complex AI and support procedures, ChatGPT proceeds to extend and the fate of this chatbot holds incredible commitment on numerous parts of our lives.

The mix of both supervised learning and support (Reinforcement) learning was utilized to tweak ChatGPT, yet the Reinforcement learning component explicitly makes ChatGPT exceptionally extraordinary. The particular strategy utilized was Reinforcement Learning from Human Feedback (RLHF), which involves human criticism in the training circle to limit hurtful, untruthful, as well as one-sided output.

ChatGPT works by endeavoring to figure out your brief, and afterward letting out series of words that it most fitting response your inquiries, in view of the information it was trained on. It was trained on humongous datasets which utilized deep neural networks to frame a profound learning brain networks demonstrated after human brain. This permits ChatGPT to learn patterns and connections in the message information to anticipate what text comes next in some random sentence.

Graphical user interface

Description automatically generated

Figure 2 - Collecting Comparison data

The training is a stage where the computer based intelligence is given guidelines and afterward given heaps of information to manage to foster its own algorithms. GPT-3 was trained on about 500 billion &quot;tokens,&quot; which permit its language models to all the more effectively dole out significance and foresee conceivable follow-on text. Many words maps to single tokens, however bigger or more mind boggling words frequently separate into various tokens. on average, tokens are about four characters in length.

However, ChatGPT does not operate at the sentence level; instead, it creates text that suggests possible words, sentences, and even paragraphs or stanzas. However, predictive text on mobile devices is not just blatantly predicting the next word; rather, it aims to produce entirely intelligible responses to any challenge. Reinforcement Learning with Human Feedback (RLHF) was used to optimize ChatGPT for discourse and improve its responsiveness to a variety of cues (RLHF).

Graphical user interface

Description automatically generated

Figure 3 - Optimize a policy

This is simply a reward model that a human constructed using comparative data so that AI could learn which response was the best.

# Model proposed

Our checker will have a user-friendly interface built using Flask, HTML, CSS, and Python where users can easily upload the file to check for plagiarism with accurate plagiarism detection instances such as text comparison, and string matching. Text preprocessing and text representation are done 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.

## 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.

Text

Description automatically generated

Figure 4 - Structure

As we can see in figure 1, we will be integrating the plagiarism checking requirements such as text preprocessing and text representation by removing the special characters, numbers, punctuations, and case conversion to keep only meaningful words. The output is forwarded to the plagiarism checker to find the text similarities between the uploaded text and the text generated by ChatGPT using cosine similarity. Based on the similarity score the result will be displayed in a graphical representation on the screen or is available for download in a file, the file will consist of the findings in a way that highlights the copied text and gives a similarity score.

## 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 |

## Subsystems

We have proposed the basic structure for our checker including I/O system, interactive system, computation system, and downloading system. For each system, there is a corresponding flow for its operation. Here are the details of them.

### I/O System

Graphical user interface, application, Teams

Description automatically generated

Figure 5 - I/O system

As we can see in figure 5, the I/O system consists of three parts. Firstly, a user should log in to the checker. After that, the user needs to upload a file that needs to be verified by our checker. Then, our checker will check the validity of the file like the extension such as txt and doc. Finally, the verified file will be passed to the next system, that is the interactive system.

### Interactive System.

Graphical user interface, application, Teams

Description automatically generated

Figure 6 - Interactive system

As we can see in figure 6, the interactive system consists of five parts. Firstly, validity checking is necessary. After that, our checker will extract keywords from the document. Then it is the communication part that will take responsibility for the socket communication between our checker and OPENAI. Finally, the return values will be stored in a folder called static. Also, these return values as parameters will be passed to the computation system.

### Computation System

Graphical user interface, application, Teams

Description automatically generated

Figure 7. Computation System

As we can see in figure 7, the computation system consists of four parts. Firstly, the computation system calculates the similarity for each file looping for ten times . After that, the highest score among all the similarities is the targeted score. Finally, the result as a parameter will be passed to the computation system.

### Downloading System

Graphical user interface, application, Teams

Description automatically generated

Figure 8 - Downloading system

As we can see in figure 8, the computation system consists of two parts. Firstly, the similarity will be displayed on the page in the form of a pie chart. After that, there is a button for user to download the checking result done by our checker.

To better understand the working flows, here is the flow chart.

Graphical user interface

Description automatically generated with low confidence

Figure 9. Flow Chart

# Experiments

For this experiment, we used ChatGPT to generate an essay with 2000 characters and then checked for plagiarism on 10 different websites. Despite using various plagiarism checking tools, COT was the only one that was able to identify the majority of the plagiarized content in the essay. Copyleaks, another plagiarism checker, was able to detect some instances of paraphrasing, but did not catch as much plagiarism as COT. The other tools that were used were unable to detect any instances of plagiarism at all.

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

Graphical user interface, chart, application, pie chart

Description automatically generated

Figure 10. COT Plagiarism results

As shown in Figure 10, COT detected majority of plagiarism as it extracts the title from input file and queries chatGPT to provide 10 different results and then computes for the similarities. COT also highlighted the matching content in the input file as shown in Figure 11.

# Conclusion

The 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 and the generated report is also available for download.

# References

1. G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. *(references)*
2. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
3. I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.
4. K. Elissa, “Title of paper if known,” unpublished.
5. R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.
6. Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE