# COT – A Plagiarism checker

## By Team-COT

Chaitra Boggaram

Oluwabusayo Omotosho

Tengjiao Zhu

Under Prof. Amr Elkady

COEN 283 - Operating Systems

Santa Clara University

### Summary

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.

For building the plagiarism checker 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 webpages. 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.

### Goals of the Project

This project can be incorporated into platforms and tools for academic and scholarly works.

The specific goal of the project is

(1) To identify instances of plagiarism in essays written using chatGPT

(2) Evaluate the degree of similarity in the given essay

(3) Provide a detailed graphical analysis of results for better viewing.

In order to know if the objectives of the project have been reached, the system should be able to identify instances of plagiarism as being truly plagiarized in each essay document i.e precision and recall.

The accuracy of the model in identifying instances of plagiarism in each written document.

The System should be user-friendly, and that would be evaluated through surveys and feedback.

### Plan of activities

Our project consists of three parts including preparation, feature engineering, model selection, building a website, and visualization of results. The details are as follows.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Stage** | **To-Do list** | **Timeline** | **Risk** | **Difficulty** |
| **Preparation** | The knowledge of Python | 2.06  |  2.12 | Studying the basic concepts might result in the delay of this stage | **MEDIUM** |
| The knowledge of Flask |
| The knowledge of Machine Learning |
| Knowledge of JavaScript and HTML |
| **Feature Engineering** | Collecting training datasets of articles | 2.13  |  2.19 | Writing a script for transforming the dataset might take more days than expectation | **HARD** |
| Feature Creation |
| Transformation |
| Feature Extraction |
| **Model Selection** | Reading the latest papers related to NLP | 2.2  |  2.26 | It is extremely difficult to have a deeper understanding of state-of-the-art papers published in top conference papers. | **HARD** |
|  |
|  |
| Pick up a model with the best performance on precision and F1-score |  |
| **Training Model** | Training the model selected on a cloud platform such as AWS | 2.2  |  2.26 | The performance of the model on the testing dataset might be much lower than the expected result, which might lead to fine-tuning the model. | **MEDIUM** |  |
|  |
|  |
|  |
| **Building an interactive website** | Create a new website with JavaScript that can take user input and can output the degree of similarity. | 2.2  |  2.26 | Debugging may lead to a delay in the delivery of the website. | **MEDIUM** |  |
|  |
|  |
|  |
| **Visualization of results** | The website will return a percentage of how similar the uploaded content is to the targeted content such as a paper and a page. | 2.27  |  3.05 |  | **MEDIUM** |  |
|  |
|  |

### Description of the final product

The final product 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.

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

<https://towardsdatascience.com/build-a-plagiarism-checker-using-machine-learning-6538110ce162>