PLAGIARISM CHECKER

Presented By: Kaushal Faldu (05)
Pushti Depani (18)

INTRODUCTION

- This project is designed to recommend similar codes based on user input.
- The program uses Natural Language Processing techniques to preprocess and vectorize the codes, and then calculates the cosine similarity between the user input and each of the codes.
- The program outputs the codes that have the highest similarity to the user input.

LIBRARIES AND MODULES USED

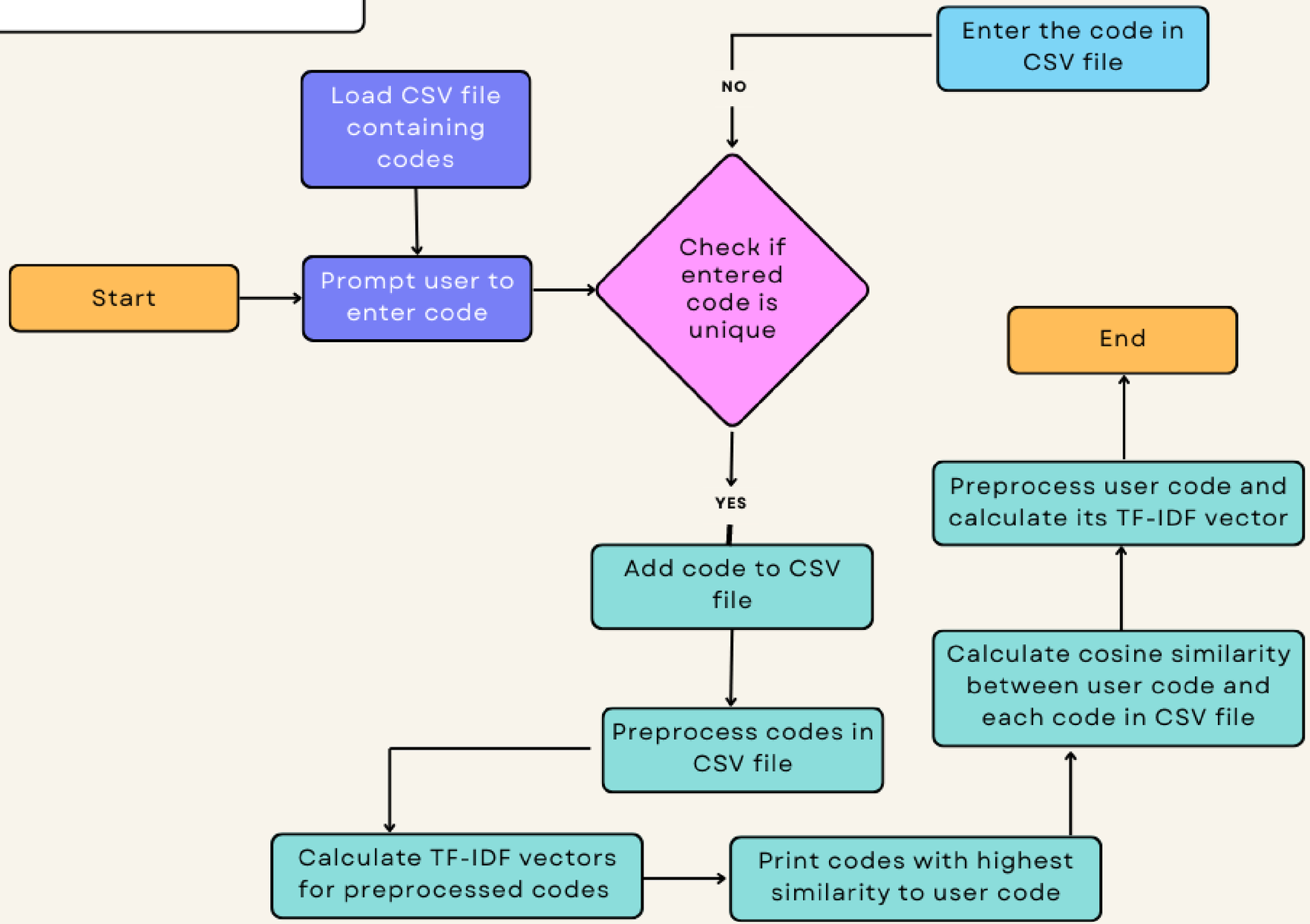
The program uses the following libraries and modules:

- csv: To read and write to CSV files.
- string: To remove punctuation from the codes.
- nltk: To preprocess the codes by removing stop words and stemming the tokens.
- sklearn: To calculate the TF-IDF vectors and cosine similarity.

PROJECT GLANCE

- This code allows a user to enter a code and finds the codes that are most similar to the entered code from a CSV file containing a list of codes.
- The code first reads the CSV file containing the codes and preprocesses them by removing punctuation, tokenizing, removing stopwords, and stemming.
- It then prompts the user to enter a code and checks if the entered code is unique.
- If it is unique, it adds the code to the CSV file and updates the list of codes.
- The code then preprocesses the codes again and calculates the TF-IDF vectors for the preprocessed codes.
- It preprocesses the user-entered code, calculates its TF-IDF vector, and calculates the cosine similarity between the user code and each of the codes. Finally, it prints the codes with the highest similarity to the user code (similarity greater than 0.8).

Flowchart


Start
Load CSV file containing codes
Prompt user to enter code
Check if entered code is unique
NO
Enter the code in CSV file
YES
Add code to CSV file
Preprocess codes in CSV file
Calculate TF-IDF vectors for preprocessed codes
Print codes with highest similarity to user code
Calculate cosine similarity between user code and each code in CSV file
Preprocess user code and calculate its TF-IDF vector
End

HOW DATASET IS CREATED

STEP - 1

Taking inputs of codes and adding them to the database.

STEP - 2

After the code is inserted we will be comparing the codes and changing the variable to '$' sign. after that all the variables will be replaced with '$'sign.

STEP - 3

Now its time to compare the logic of the code and give the plagiarism which is obtained and the similarity. So for that we are using fuzzywuzzy liberary to compare the codes and then the final output will be stored in the dataset.

IMPLEMENTATION

Using Artificial Intelligence creating plagiarism checker for the codes. Where similar codes logic will be detected and plagiarism report will be generated at the end. We are using string matching algorithm and Natural Language Processing to detect plagiarism.

KEY INDICATOR	Similar variables will be detected.	After that we will be changing all the variables by any symbol. And ignore the keywords.	At last we will be comparing the codes, as only the logic will be compared and giving the report.
USE/ AIM / ADVANTAGE	The main aim behind this is the codes will not get copied and if they are then they wil be plagiarised.	Will get some new logic in every definition as all the codes will be plagiarized and checked.	Allows you to check if you correctly identified the logic you used

EXPLANATION OF PROJECT

- This code performs a content-based similarity analysis between a user-entered code and a collection of codes stored in a CSV file. The CSV file is loaded into memory, and each code is preprocessed to remove stop words, perform stemming, and transform the text into a TF-IDF vector representation.
- When the user enters a new code, the code checks if it is unique by comparing it with the existing codes. If it is unique, the code is added to the CSV file, and the list of codes is updated. Then, the new code is preprocessed and transformed into a TF-IDF vector representation.
- Next, the cosine similarity between the user code and each of the codes is calculated using the TF-IDF vectors. Finally, codes with a similarity score above 0.8 are printed, along with their similarity scores, indicating that they are similar to the user-entered code.
- The preprocessing steps include removing punctuation, tokenizing the text, removing stop words, stemming the tokens, and joining them back into a single string. These steps help to standardize the text and reduce the noise in the data, making it easier to compare the codes.
- Overall, this code provides a simple example of how content-based similarity analysis can be performed using TF-IDF vectors and cosine similarity. However, it is worth noting that there are many other techniques and considerations when performing similarity analysis, such as using word embeddings or considering the context and syntax of the code.

CONCLUSION AND FUTURE WORK

The conclusion that can be made from this code is that it can be used to find codes that are similar to a user-entered code by calculating the cosine similarity between their TF-IDF vectors. This can be useful in various applications such as plagiarism detection, code recommendation, and code similarity analysis. Additionally, the code demonstrates how to read and write CSV files in Python, preprocess text data using the NLTK library, and use the scikit-learn library to calculate TF-IDF vectors and cosine similarity.

The future work which we will be doing is that we will doing is that we will be creating proper page using flask and everything will be displayed and the percentage and the comparison will be shown on the same page and also we will be able to download the report of the plagiarism.