PLAGARISM CHECKER

A PROJECT REPORT

Submitted by

ABISHEK .S (202109005)
KARNAS SAGAR S (202109026)
MOHAMED ASLAM K (202109034)

In partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

MEPCO SCHLENK COLLEGE OF ENGINEERING,
SIVAKASI-626123

(An Autonomous Institution)

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2023

MEPCO SCHLENK ENGINEERING COLLEGE, SIVAKASI

AUTONOMOUS

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

BONAFIDE CERTIFICATE

Certified that this project report "**PLAGARISM CHECKER**" is the bonafide work of **MOHAMED ASLAM.K (202109034), ABISHEK.S (202109005), KARNAS SAGAR.S (202109026)** who carried out the project work under my supervision.

SIGNATURE

Dr. A.Shenbagarajan, M.E.,Ph.D
Dr.P.Thendral,M.E.,Ph.D
Assistant Professor(SG)
Artificial Intelligence and Data Science
Mepco Schlenk Engineering College

SIGNATURE

Dr.J.Angela Jennifa Sujana, M.E.,Ph.D
Associate Professor(SG) & Head
Artificial Intelligence and Data Science
Mepco Schlenk Engineering College,
Sivakasi- 626 005,Virudhunagar

The project report submitted for the viva voce held on …….

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

First and foremost we **praise and thank "The Almighty",** the lord of all creations, who by his abundant grace has sustained us and helped us to work on this project successfully.

We really find unique pleasure and immense gratitude in thanking our respected management members**,** who is the backbone of our college.

A deep bouquet of thanks to respected Principal **Dr.S.Arivazhagan M.E.,Ph.D.,** for having provided the facilities required for our mini project.

We sincerely thank our Head of the Department **Dr. J. Angela Jennifa Sujana M.E.,Ph.D.,** Associate Professor(SG) & Head, Department of Artificial Intelligence and Data Science, for her guidance and support throughout the mini project .

We also thank our guide **Dr.A.Shenbagarajan.,M.E.,Ph.D.,** Assistant Professor(SG), **Dr.P.Thendral,M.E.,Ph.D.,** Assistant Professor Department of Artificial Intelligence and Data Science for their valuable guidance and it is great privilege to express our gratitude to them.

We extremely thank project coordinator **Dr.A.Shenbagarajan.,M.E.,Ph.D.,** Assistant Professor(SG), **Dr.P.Thendral,M.E.,Ph.D.,** Assistant Professor Department of Artificial Intelligence and Data Science, who inspired us and supported us throughout the mini project.

We extend our heartfelt thanks and profound gratitude to all the faculty members of Artificial Intelligence and Data Science department for their kind help during our mini project work.

We also thank our parents and our friends who had been providing us with constant support during the course of the mini project work.

TABLE OF CONTENTS

ABSTRACT

This code presents a plagiarism checker tool that utilizes artificial intelligence and data analytics techniques to analyze text similarity. The code is built using Python libraries such as `sklearn`, `tkinter`, `PIL`, and `matplotlib` to create a graphical user interface (GUI) and perform advanced text analysis.

The algorithm behind the tool includes functions for converting text into numerical feature vectors, calculating cosine similarity, and computing plagiarism scores. The code's GUI allows users to select a text file for analysis and presents the results through intuitive visualizations, including bar graphs, pie charts, and line plots.

The tool's effectiveness lies in its use of TF-IDF vectorization and cosine similarity measures, which accurately assess the similarity between texts and detect potential cases of plagiarism. The visualizations further aid in the interpretation of the results, providing users with a comprehensive understanding of the detected plagiarism levels.

In summary, this code offers a powerful plagiarism checker that combines artificial intelligence and data analytics techniques to provide an efficient and user-friendly solution. It has the potential to be a valuable tool for educational institutions, content creators, and researchers, enabling them to identify and address instances of textual plagiarism effectively.

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

A plagiarism checker uses advanced database software to scan for matches between your text and existing texts. They are used by universities to scan student assignments. There are also commercial plagiarism checkers you can use to check your own work before submitting.

Behind the scenes, plagiarism checkers crawl web content and index it, scanning your text for similarities against a database of existing content on the internet. Exact matches are highlighted using keyword analysis. Some checkers can also identify non-exact matches (paraphrasing plagiarism).

On the user end, the checker typically provides you with a plagiarism percentage, highlights the plagiarism, and lists the sources. You can get an interactive look at the Scribbr Plagiarism Checker below. Not every plagiarism checker has access to the same database. This can lead to major differences in results.

Free plagiarism checkers often have smaller databases. This means that there are large gaps in their ability to find matches, especially with less readily available online content. The highest-quality plagiarism checkers have larger databases, enhancing their ability to find matches.

This can range from re-submitting an entire assignment to reusing passages or data from something you've turned in previously without citing them.
Paraphrasing without citation is the most common type of plagiarism. Paraphrasing, like quoting, is a legitimate way to incorporate the ideas of others into your writing. It only becomes plagiarism when you rewrite a source's points as if they were your own Plagiarism can be accidental or intentional. Copying an entire essay or story and calling it your own is plagiarism. Copying one sentence word-for-word without "quotations" is also plagiarism. Whether you hand it in to a teacher, or post it in your blog, plagiarism is against the law in most nations.

1.1.SCOPE OF THE PROJECT

If an essay or dissertation builds on previous work, it is essential that this is clearly identified in the text and is appropriately referenced, as if it were written by a different person. The assessors should be in no doubt as to what work the student has completed in their current degree course and it is this that will be assessed.

When submitting coursework, students will be asked to declare that no part of their work has already been submitted, or is being submitted, for any other qualification.

1.2.OBJECTIVE OF THE PROJECT

A plagiarism checker uses advanced database software to scan for matches between your text and existing texts. They are used by universities to scan student assignments. There are also commercial plagiarism checkers you can use to check your own work before submitting. Plagiarism may result in a rejection of ongoing applications and more importantly, in subsequent long-term non-eligibility for research funding by these applicants. Also journal editors rely more and more on plagiarism detection software before accepting manuscrits for publication.

1.3.REPORT SUMMARY

Plagiarism checkers are valuable tools that can be used for various purposes, avoiding duplicate content penalties from search engines, and providing valuable and authentic information to readers.They can identify unintentional instances of plagiarism or incomplete citations, allowing writers to make the necessary corrections.Plagiarism checkers are valuable tools for educators and instructors to educate students about proper citation practices and academic integrity.

CHAPTER 2

MODULES

2.1.INTRODUCTION

Scikit-learn is a popular Python library used for machine learning tasks. It provides a wide range of tools and algorithms for tasks such as classification, regression, clustering, and dimensionality reduction. Scikit-learn is built on top of other scientific computing libraries in Python, such as NumPy, SciPy, and matplotlib, and is designed to be easy to use and integrate with other libraries in the Python ecosystem.

2.1. SCIKIT LEARN

Scikit-learn is widely used in the industry and academia for building machine learning models, and it's particularly popular for its ease of use and extensive documentation. It provides a range of tools for data preprocessing, feature extraction, and model selection, as In scikit-learn, feature extraction refers to the process of transforming raw data into a format that can be effectively used by machine learning algorithms. scikit-learn provides various techniques for feature extraction, including preprocessing, dimensionality reduction, and feature selection.

2.1.1. FEATURE EXTRACTION

TfidfVectorizer:

TfidfVectorizer can be used to preprocess and represent the textual content of documents as numerical vectors. Here's a general workflow for using TfidfVectorizer in a plagiarism checker:

1. Data Preparation: Gather the documents or texts that you want to compare for plagiarism detection.
2. Preprocessing: Clean and preprocess the text data to remove irrelevant information, such as punctuation, stopwords, and convert the text to lowercase. You can use techniques like tokenization, stemming, and lemmatization.

3.Vectorization: Instantiate the TfidfVectorizer class and fit it on your preprocessed text corpus. This step involves creating a vocabulary of unique words (or n-grams) present in the corpus and computing the TF-IDF values for each word in each document

2.1.2. METRICS

Scikit-learn is a popular machine learning library in Python that provides a wide range of tools and functionalities for various tasks, including model evaluation. Scikit-learn offers several metrics to assess the performance of machine learning models.

Cosine Similarity:

The cosine similarity is a common metric used in plagiarism checkers to measure the similarity between two pieces of text. It determines the similarity based on the angle between the vectors representing the texts in a high-dimensional space. Here's a basic overview of how cosine similarity is used in a plagiarism checker:

Cosine Similarity Calculation: Once the texts are vectorized, the cosine similarity is calculated using the formula:

cosine_similarity(A, B) = (A · B) / (||A|| * ||B||)

1. where A and B are the vector representations of the texts, · represents the dot product, and ||A|| and ||B|| represent the Euclidean norms of A and B, respectively.

2. Threshold Comparison: The cosine similarity value is compared against a predefined threshold to determine whether the texts are similar or not. The threshold depends on the specific plagiarism detection system and can vary.

2.3.GRAPHICAL USER INTERFACE

A graphical user interface (GUI) is an interface that is drawn on the screen for the user to interact with. User interfaces have some common components: Main window. Menu.Fig 2.1 shows the Graphical User Interface for the source code mentioned bellow

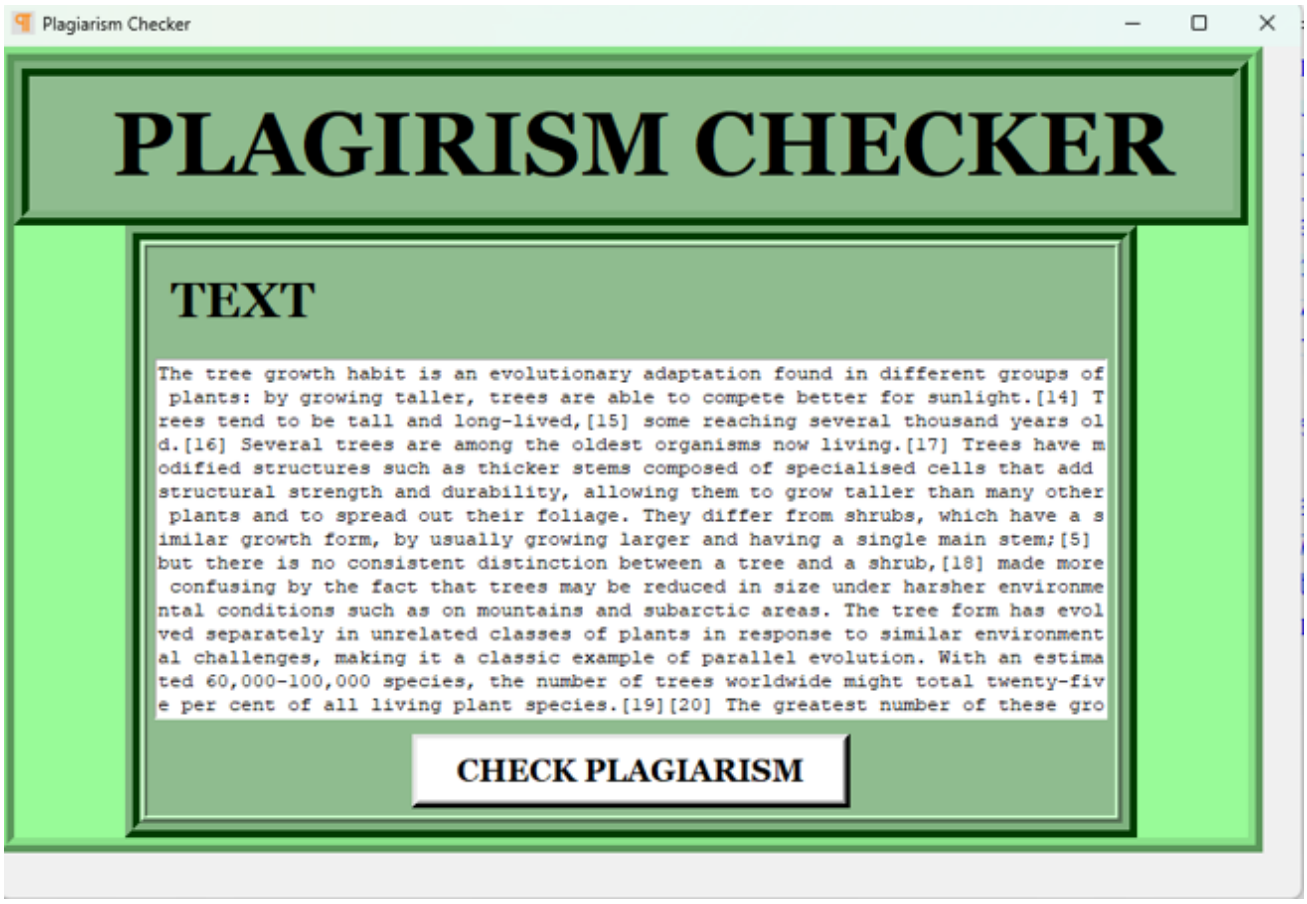

Figure. 2.1 Graphical User Interface

TKINTER

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

To create Tabs in the window application. Tabs are generally used to separate the workspace and specialize the group of operations in applications at the same time.

FILE DIALOUGE BOX:

The filedialog module in the tkinter library provides a convenient way to create file dialog boxes in Python GUI applications. These dialog boxes allow users to interactively select files from their system.

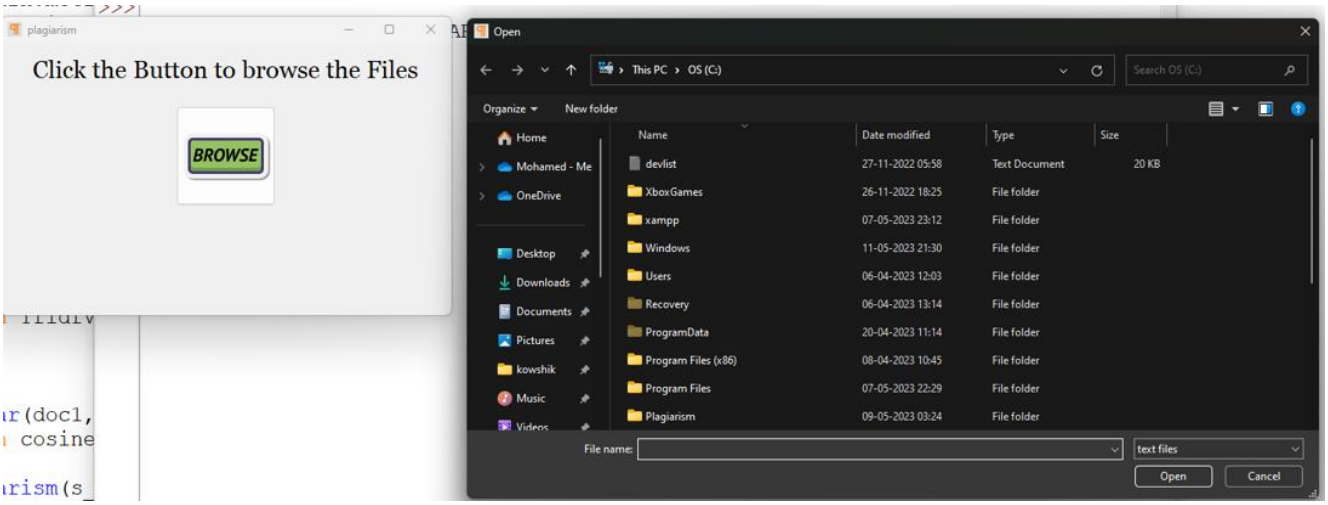

Figure 2.2 File Dialog Box

Figure. 2.2 is the image of the File Dialog Box widget for the source code

MESSAGE BOX:

The messagebox module in the tkinter library provides a simple way to display message boxes or dialogs in Python GUI applications. These message boxes are useful for displaying information, warnings, errors, or asking for user confirmation.

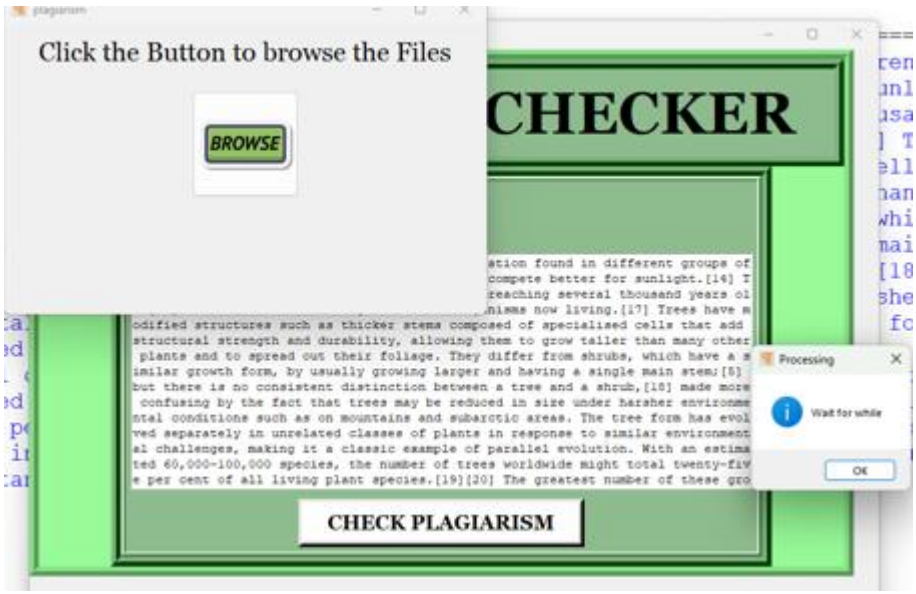

Figure 2.3 Message Box

Figure 2.3 is the Image of the Message Box for the Source

CHAPTER 3

IMPLEMENTATION

3.1. INTRODUCTION

This code appears to be a plagiarism checker implemented using Python and various libraries such as scikit-learn (sklearn), tkinter, matplotlib, and PIL (Python Imaging Library). It includes a graphical user interface (GUI) that allows the user to browse and select a text file, choose a language, and perform plagiarism checking.

3.1.1 ARITIFICIAL INTELLIGENCE CONCEPT

The artificial intelligence (AI) concept in the provided code lies in the utilization of the TF-IDF (Term Frequency-Inverse Document Frequency) vectorization and cosine similarity calculation.

1. **TF-IDF Vectorization:** The TfidfVectorizer from scikit-learn is used to convert the text data into numerical representations called TF-IDF vectors. TF-IDF assigns weights to words based on their frequency in a specific document and their rarity across all documents. By using TF-IDF vectorization, the code transforms the textual content into a numerical format that can be processed and analyzed by machine learning algorithms.

2. **Cosine Similarity:** The cosine_similarity function from scikit-learn's metrics.pairwise module is employed to calculate the cosine similarity between the TF-IDF vectors of different documents. Cosine similarity measures the similarity between two vectors by calculating the cosine of the angle between them. In the context of plagiarism detection, higher cosine similarity values indicate a higher level of similarity between documents.

These AI techniques allow the code to analyze and compare the textual content of multiple documents to determine their similarity and identify potential instances of plagiarism. By leveraging TF-IDF vectorization and cosine similarity, the code can automate the process of checking for plagiarized content, enabling efficient and effective plagiarism detection.

3.1.2 DATA ANALYTICS CONCEPT

The data analytics concept in the provided code can be observed in the following aspects:

1. **Similarity Analysis:** The code utilizes cosine similarity, a technique commonly used in data analytics, to measure the similarity between TF-IDF vectors. By calculating the cosine similarity between the TF-IDF vector of a selected file and the TF-IDF vectors of other files, the code determines the degree of similarity between the texts. This similarity analysis is a fundamental concept in data analytics, where finding patterns, relationships, or similarities in data plays a crucial role.

2. **Data Visualization:** The code generates various visualizations using the matplotlib library to represent the plagiarism levels. It creates bar charts, pie charts, line plots, and filled plots to visualize the plagiarism scores and provide a graphical representation of the data. Data visualization is a crucial aspect of data analytics as it helps in gaining insights, identifying patterns, and presenting the analysis results in a more intuitive and understandable manner.

By incorporating these data analytics concepts, the code enables the analysis and visualization of textual data to identify plagiarism and provide meaningful insights about the similarity levels among different documents.

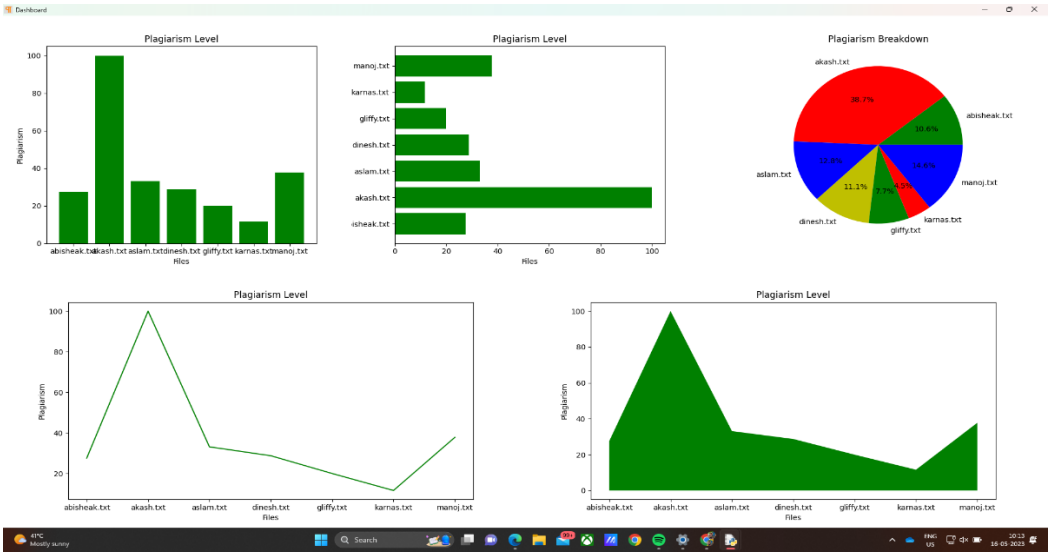

Figure 3.1 Visulazing The Results of the similarity score

3.2 ALGORITHM

Import the necessary libraries and modules, including TfidfVectorizer, os, cosine_similarity, tkinter, time, sleep, Image, ImageTk, tkinter.messagebox, and matplotlib.pyplot.

1. Define a function named convert_array that takes a text as input and returns the TF-IDF vector representation of the text.

2. Define a function named similar that takes two documents as input and calculates the cosine similarity between them.

3. Define a function named plagiarism that takes two sets of document vectors and a list of files as input. It calculates the similarity score between each document and a reference document, stores the results in a dictionary, and plots various graphs using matplotlib.

4. Define a function named check_plagiarism that takes a value from a file and a language as input. It retrieves files from a specific directory, reads the contents of each file, appends the value from the file to the list of notes, converts the notes to vectors, and calls the plagiarism function.

5. Define a function named file_insert that prompts the user to select a file, reads the contents of the file, and creates a GUI window using tkinter to display the text and a button to check plagiarism.

6. Create a root window using tkinter and set its title and size.

7. Create a label and a button in the root window using tkinter to browse the files.

8. Run the tkinter event loop to start the application.

9. End the program.

3.3 BLOCK DIAGRAM

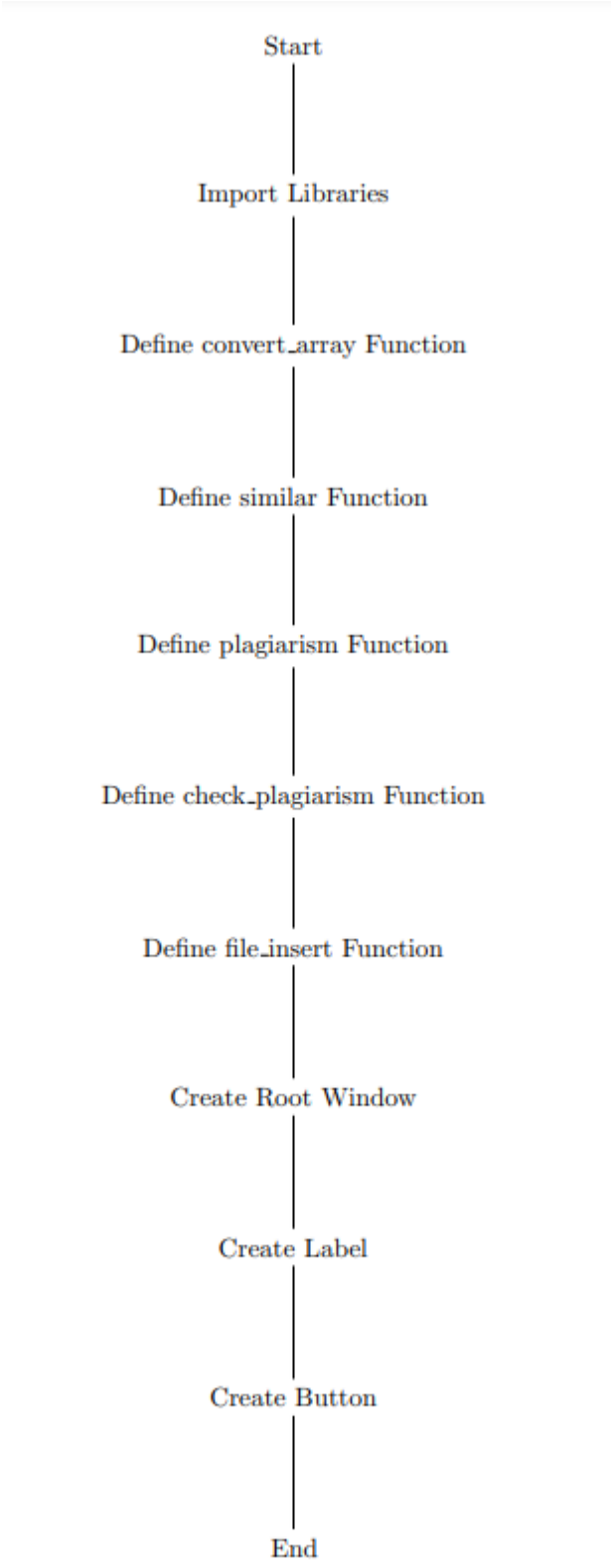

Figure 3.2 Flow of the Program

CHAPTER 4

SOURCE CODE

```
from sklearn.feature_extraction.text import TfidfVectorizer
import os
from sklearn.metrics.pairwise import cosine_similarity
from tkinter import*
import tkinter as tk
from tkinter import ttk
from tkinter import filedialog as fd
from time import time, sleep
from PIL import Image, ImageTk
import tkinter.messagebox
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import matplotlib.pyplot as plt

def convert_array(text):
    return
TfidfVectorizer(strip_accents='unicode').fit_transform(text).toarray
()
def similar(doc1,doc2):
    return cosine_similarity([doc1,doc2])

def plagiarism(s_vector1,s_vector2,files):
    ans_dict={}
    plagiarism_res=set()
    print(s_vector1)
    print(s_vector2)
    text_vector_a=s_vector2
    i=0
    for text_vector_b in s_vector1:
```

```
        sim_score=similar(text_vector_a,text_vector_b)[0][1]
        score=sim_score
        plagiarism_res.add(score)
    for k in plagiarism_res:
        if i<len(files):
            ans_dict[files[i]]=(k*100)
            i+=1
    print(plagiarism_res)

plt.rcParams["axes.prop_cycle"]=plt.cycler(color=['g','r','b','y'])
    fig1,ax1=plt.subplots()
    ax1.bar(ans_dict.keys(),ans_dict.values())
    ax1.set_title("Plagiarism Level")
    ax1.set_xlabel("Files")
    ax1.set_ylabel("Plagiarism")
    fig2,ax2=plt.subplots()
    ax2.barh(list(ans_dict.keys()),ans_dict.values())
    ax2.set_title("Plagiarism Level")
    ax2.set_xlabel("Files")
    ax2.set_ylabel("Plagiarism")
    fig3,ax3=plt.subplots()

ax3.pie(ans_dict.values(),labels=ans_dict.keys(),autopct='%1.1f%%')
    ax3.set_title("Plagiarism Breakdown")
    fig4,ax4=plt.subplots()
    ax4.plot(list(ans_dict.keys()),list(ans_dict.values()))
    ax4.set_title("Plagiarism Level")
    ax4.set_xlabel("Files")
    ax4.set_ylabel("Plagiarism")
    fig5,ax5=plt.subplots()
```

```
    ax5.fill_between(ans_dict.keys(),ans_dict.values())
    ax5.set_title("Plagiarism Level")
    ax5.set_xlabel("Files")
    ax5.set_ylabel("Plagiarism")
    root3=Tk()
    root3.title("Dashboard")
    root3.iconbitmap('A:/Plagiarism Project/icon.ico')
    root3.state('zoomed')
    upper_frame=Frame(root3)
    upper_frame.pack(fill="both",expand=True)
    canvas1=FigureCanvasTkAgg(fig1,upper_frame)
    canvas1.draw()

canvas1.get_tk_widget().pack(side="left",fill="both",expand=True)
    canvas2=FigureCanvasTkAgg(fig2,upper_frame)
    canvas2.draw()

canvas2.get_tk_widget().pack(side="left",fill="both",expand=True)
    canvas3=FigureCanvasTkAgg(fig3,upper_frame)
    canvas3.draw()

canvas3.get_tk_widget().pack(side="left",fill="both",expand=True)
    lower_frame=Frame(root3)
    lower_frame.pack(fill="both",expand=True)
    canvas4=FigureCanvasTkAgg(fig4,lower_frame)
    canvas4.draw()

canvas4.get_tk_widget().pack(side="left",fill="both",expand=True)
    canvas5=FigureCanvasTkAgg(fig5,lower_frame)
    canvas5.draw()
```

```
canvas5.get_tk_widget().pack(side="left",fill="both",expand=True)

def check_plagiarism(Val_from_file,Lan):
        tkinter.messagebox.showinfo("Processing","Wait for while")
        sleep(5)
        os.chdir('C:\Plagiarism\English')
        files=[i for i in os.listdir() if i.endswith('.txt')]
        notes=[open(_file,encoding='utf-8').read() for _file in
files]
        notes.append(Val_from_file)
        vectors1=convert_array(notes)
        vectors2=vectors1[(len(vectors1)-1)]
        plagiarism(vectors1,vectors2,files)

def file_insert():
    filetypes = (('text files', '*.txt'),
                 ('All files', '*.*'))
    f = fd.askopenfile(filetypes=filetypes,
                       initialdir="C:\\")
    if f is None:
        tkinter.messagebox.showerror("ERROR","NO FILE PATH")
    else:
        st=""
        for i in f.readlines():
            if i !='{' or i!='}':
                st+=i
        print(st)
        new_root = Tk()
        new_root.title("Plagiarism Checker")
```

```
        new_root.iconbitmap('A:/Plagiarism Project/icon.ico')
        new_root.geometry("1350x800")
        Tab1 = Frame(new_root, bd=10,bg="pale green",width=1350,
height=700,relief=RIDGE)
        Tab1.grid()
        topframe1=Frame(Tab1, bd=10,bg="dark green",width=1340,
height=100, relief=RIDGE)
        topframe1.grid()
        center_frame=Frame(Tab1,bd=10,bg="dark
green",width=800,height=600,relief=RIDGE)
        center_frame.grid()
        centre_mid=Frame(center_frame,bd=5,width=1330,bg="dark sea
green",height=550,relief=RIDGE)
        centre_mid.grid()
        centre_mid1=Frame(centre_mid,bd=5,width=1320,bg="dark sea
green",height=80)
        centre_mid1.grid(sticky=W)
        centre_mid2=Frame(centre_mid,bd=5,width=1310,bg="dark sea
green",height=80)
        centre_mid2.grid(sticky=W)
        centre_mid4=Frame(centre_mid,bd=5,width=1300,bg="dark sea
green",height=80)
        centre_mid4.grid()
        title=Label(topframe1,font=('Georgia',45,'bold'),bg="dark
sea green",text="   PLAGIRISM CHECKER  ",bd=10)
        title.grid(row=0,column=0,sticky=W)
        title=Label(centre_mid1,font=('Georgia',25,'bold'),bg="dark
sea green",text="TEXT",bd=10)
        title.grid(row=0,column=0,sticky=W)
        text_box=Text(centre_mid2,height=15)
```

```
        text_box.grid(row=0,column=0,sticky=W)
        text_box.insert('15.0',st)

but1=Button(centre_mid4,bd=5,font=('Georgia',16,'bold'),width=20,bg=
"white",text="CHECK
PLAGIARISM",command=lambda:check_plagiarism(st,radiobut.get()))
        but1.grid(row=0,column=0)
# root window
root = tk.Tk()
radiobut=StringVar()
root.title('plagiarism')
root.iconbitmap('A:/Plagiarism Project/icon.ico')
root.geometry('500x300')
label = Label(root,text="Click the Button to browse the Files",
font=('Georgia 20'))
label.pack(pady=10)
photo = PhotoImage(file = r"A:\Plagiarism Project\buttons.png")
# Create a Button
ttk.Button(root,
text="Browse",image=photo,command=file_insert).pack(pady=12)
root.mainloop()
```

RESULTS

Figure 5.1 Importing the file

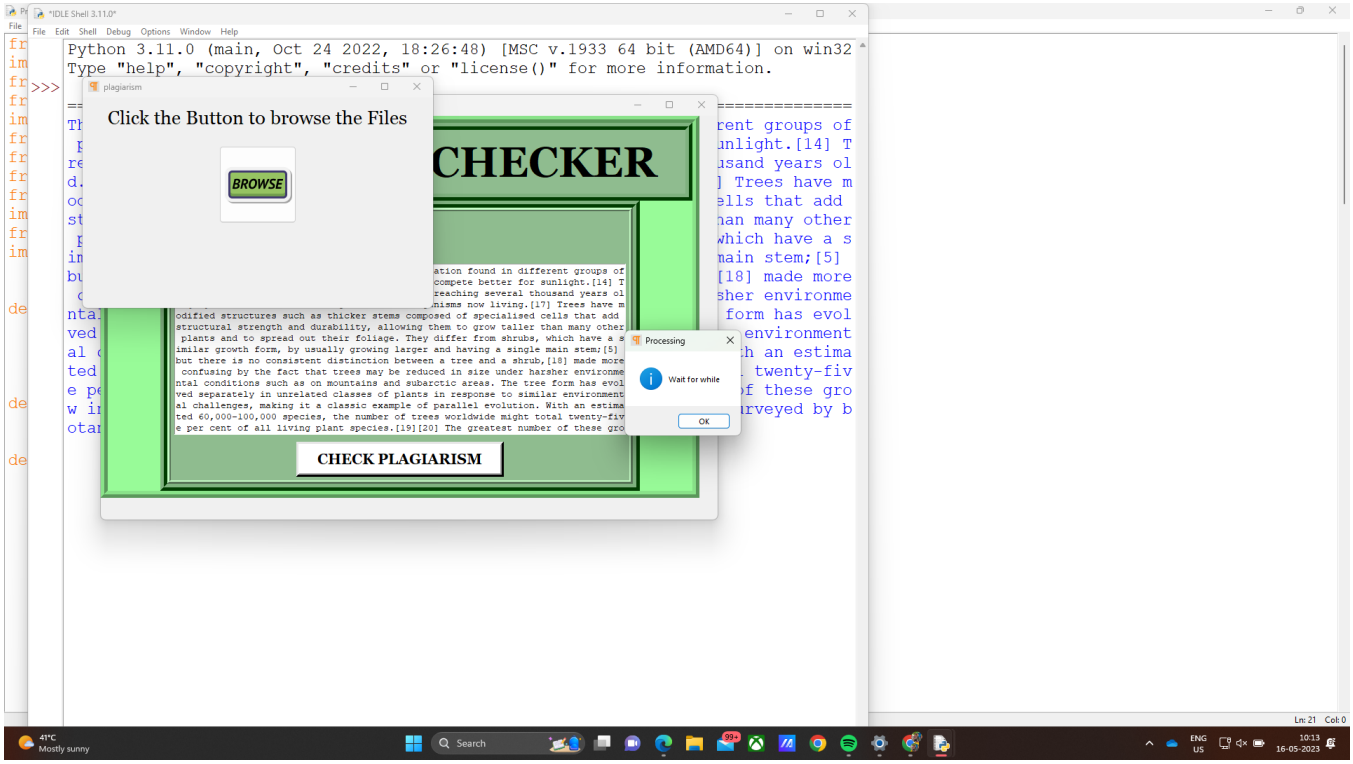

Figure 5.2 Processing The results

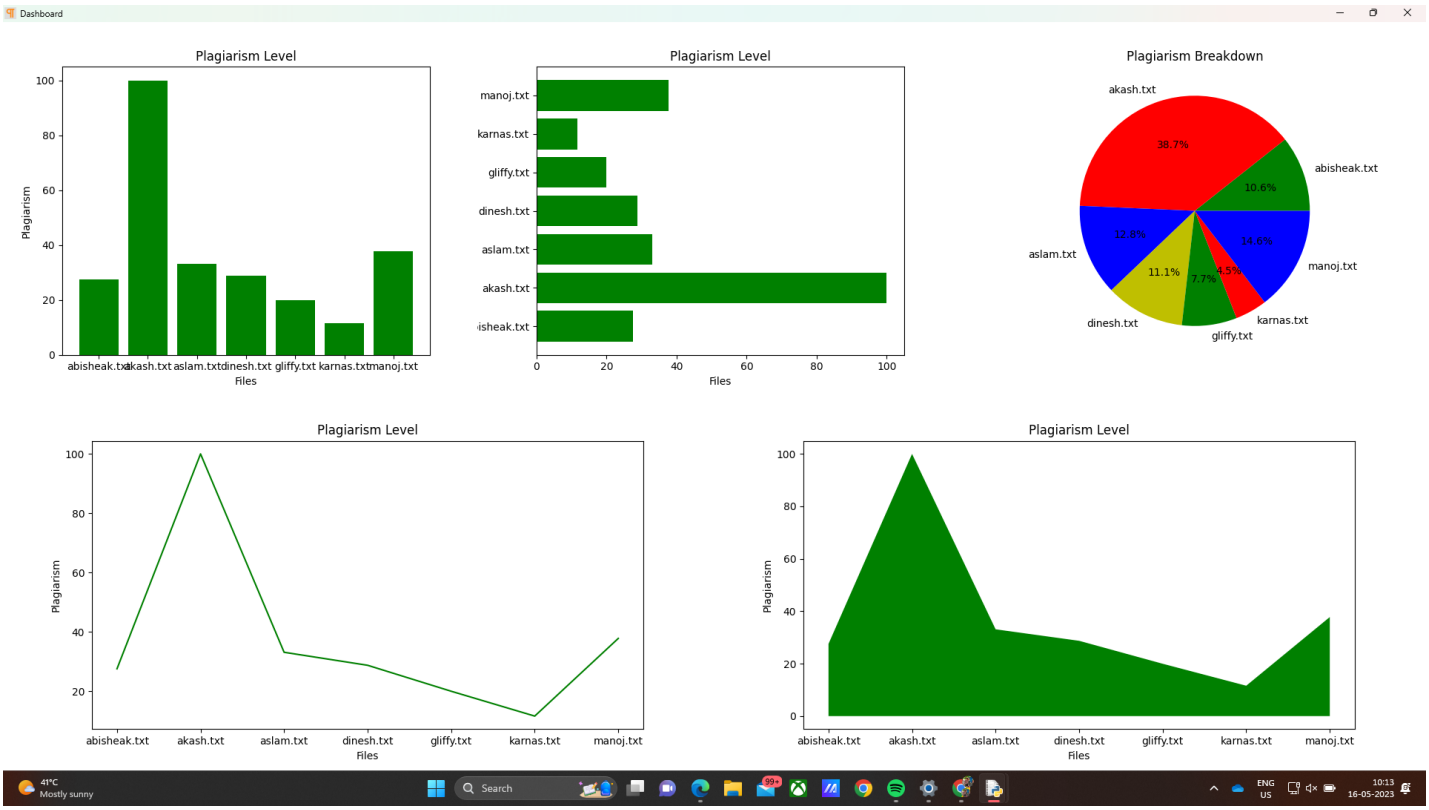

Figure 5.3 Results

CONCLUSION

The provided code implements a plagiarism checker tool using natural language processing and data analytics concepts. It utilizes libraries such as sklearn, tkinter, PIL, and matplotlib to create a GUI and perform plagiarism analysis.

The algorithm includes functions for converting text into numerical feature vectors, calculating cosine similarity, and computing plagiarism scores. The code's GUI allows users to select a text file for analysis and presents the results through visualizations like bar graphs, pie charts, and line plots.

In conclusion, the code offers a functional plagiarism checker tool with a user-friendly interface. It uses TF-IDF vectorization and cosine similarity to measure similarity between texts, helping detect potential plagiarism. The code's visualizations enhance result interpretation. Overall, it serves as a valuable tool for identifying textual plagiarism and can be customized as needed.