جامعة زمزم للعلوم Zamzam University والتكنولوجيا Of science and



Technology

“Connects learning to real life”

**Plagiarism Detection System**

Group Members:-

|  |  |  |
| --- | --- | --- |
| NO. | ID | Name |
|  | CIS220103 | Abdimalik Abdirahman Mohamed |
|  | CIS220079 | Abdullahi Abukar Omar |
|  | CIS220149 | Abdullahi Muhidin Omar |
|  | CIS220147 | Abdishakur Anshur Abdulle |
|  | CIS220099 | Mohamed Omar Abdisalam |
|  | CIS220253 | Asha Hussein Ahmed |
|  | CIS20040 | Omar Abdullahi Ahmed |
|  | CIS220348 | Naima Omar Wasuge |
|  | CIS220242 | Sundus Ismail Ali |
|  | CIS220288 | Abdihakin Abdullahi Muhidin |

**Table of Contents**

1. Introduction …………………………………………………….1

* Purpose of the system ………………………………….1
* Key features……………………………………………...1

1. Problem Statement…………………………………………….1

* What This Project Solves……………………………...1
* Importance of Detecting Plagiarism…………………..2

1. System requirements ………………………………………….2

* Hardware requirements………………………………….2
* Software requirements…………………………………..2
* Dependencies ……………………………………………2

1. How to use the system………………………………………...4

* Uploading files………………………………………...….4
* Generating Similarity Reports…………………………..5
* Interpreting the Results………………………………….5

1. File formats and compatibility………………………………...6

* Supported file types……………………………………...6
* Handling multiple files…………………………………...6

1. Features overview……………………………………………...6

* Sentence tokenization……………………………………6
* Cosine similarity calculation……………………………..7
* Interactive graphs (Scatter, Bar, Histogram, violin )…..7

1. Code explanation……………………………………………….8

* Streamlit interface………………………………………...8
* Core functions……………………………………………..8
* Similarity calculations……………………………………..9

1. Acknowledgments ………………………………………………9
2. **Introduction**

Plagiarism is the act of using someone else’s words, ideas, or work without proper acknowledgment, presenting it as your own. It includes copying text, or using creative works like code without permission.

**Purpose of the system**

Its main purpose is to help users to determine whether a publication is similar to another article or publication, thus providing an easier way to detect suspected plagiarism. Such a system is advantageous for students, teachers, researchers, and content creators.

**Key Features**

It has support for many different file types :. txt,. pdf,. docx

Remove the output of the cosine similarity extractions

Interactive visualizations such as scatter plots, bar plots, histograms and violin plots

Streamlit powered User-Friendly Interface

1. **Problem Statement**

**What This Project Solves**

Plagiarism detection is essential in many areas, particularly in academia and content creation. Manually spotting similarities between documents can be tedious and prone to mistakes. This project streamlines the process, offering a dependable and efficient solution for:

* Ensuring originality in academic submissions
* Preventing duplicate content in professional environments
* Promoting ethical research practices

**Importance of Detecting Plagiarism**

Identifying plagiarism encourages intellectual integrity and ensures adherence to copyright and ethical guidelines. This tool makes the detection process easier, making it available to everyone.

1. **System requirements**

**Hardware Requirements**

Minimum: 4GB RAM, 2 GHz dual-core processor

Recommended: 8GB RAM, 2.5 GHz quad-core processor

**Software Requirements**

Operating System: Windows, macOS, or Linux

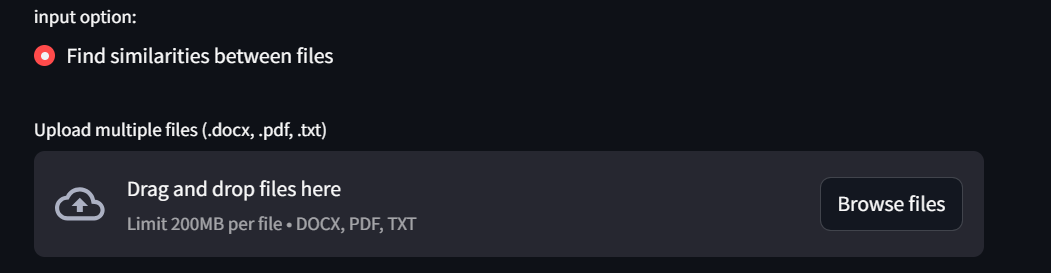
Python version 3.7 or higher.

#### Dependencies

1. **import streamlit as st**
   * **Purpose:** Streamlit is a library used for creating interactive web apps in Python.
   * **as st:** This is an alias. Instead of writing streamlit every time, you can simply write st.
   * **You can install** (pip install streamlit).
2. **import pandas as pd**
   * **Purpose:** Pandas is used for working with tables or datasets (DataFrames). It allows easy data analysis and manipulation.
   * **as pd:** Another alias to shorten the name of the library.
   * **You can install** (pip install pandas).
3. **import nltk**
   * **Purpose:** Natural Language Toolkit (nltk) is used for processing and analyzing human language data (like text).
   * **nltk.download('punkt'):** Downloads a tokenizer called punkt, which splits text into sentences or words.
   * **You can install** (pip install pandas).
4. **from nltk import tokenize**
   * **Purpose:** Imports the tokenize module from nltk, specifically for splitting text into smaller parts, like sentences or words.
5. **from bs4 import BeautifulSoup**
   * **Purpose:** BeautifulSoup is used for scraping data from websites (HTML content). It helps extract specific elements (like paragraphs or links).
   * **You can install** (pip install beautifulsoup4).
6. **import requests**
   * **Purpose:** Requests is used to send HTTP requests (like visiting a website or getting its content). In this case, it's used to fetch the text of web pages.
7. **from sklearn.feature\_extraction.text import CountVectorizer**
   * **Purpose:** CountVectorizer converts text into numerical data for analysis. It helps count the frequency of words in a text.
8. **from sklearn.metrics.pairwise import cosine\_similarity**
   * **Purpose:** cosine\_similarity calculates how similar two pieces of text are. It's a mathematical way to measure similarity between two documents.
   * **You can install** (pip install scikit-learn).
9. **import io**
   * **Purpose:** The io module provides tools for handling file input/output operations. It's used here for reading files (like text files).
10. **import docx2txt**
    * **Purpose:** This library is used to extract text from Microsoft Word documents (.docx files).
    * **You can install** (pip install docx2txt).
11. **from PyPDF2 import PdfReader**
    * **Purpose:** PdfReader is used to read and extract text from PDF files.
    * **You can install** (pip install pypdf2).
12. **import plotly.express as px**
    * **Purpose:** Plotly Express is a visualization library. It creates interactive charts like scatter plots, bar charts, and histograms.
    * **You can install** (pip install plotly).
13. **How to use the system**

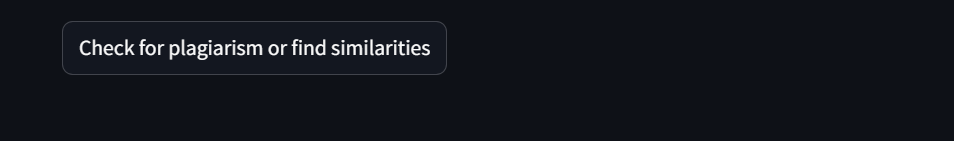
#### Uploading Files

* Use the "Upload files" option to upload .txt, .pdf, or .docx files.



#### Generating Similarity Reports

* Click on the "Check for plagiarism or find similarities" button to generate a similarity report.



#### Interpreting the Results

* View similarity percentages and interactive graphs to understand overlaps between files.



1. **File formats and compatibility**

#### Supported File Types

* Plain Text (.txt)
* PDF (.pdf)
* Word Documents (.docx)

#### Handling Multiple Files

* Upload multiple files simultaneously for batch processing and detailed similarity reports.

1. **Features overview**

#### Sentence Tokenization

This feature analyzes the uploaded text by breaking it down into individual sentences with the help of the NLTK library. By using sentence tokenization, it enables a detailed examination and helps identify precise areas of similarity between different texts.

Description: This functionality is beneficial for detecting particular sections of content that might have been copied.

Use Case: It is particularly useful for recognizing specific parts of text that may have been plagiarized.

We have used this library and function :

from nltk import tokenize

*def* get\_sentences(*text*):

    sentences = tokenize.sent\_tokenize(text)

    return sentences

**Cosine similarity calculation**

#### Interactive Graphs

**Scatter Plots**: Displays pairwise similarities between files, making it easy to identify highly similar pairs visually.

*def* plot\_scatter(*df*):

    fig = px.scatter(df, *x*='File 1', *y*='File 2', *color*='Similarity', *title*='Similarity Scatter Plot')

    st.plotly\_chart(fig, *use\_container\_width*=True)

**Bar Charts**:Provides a comparative view of similarity scores between files.

*def* plot\_bar(*df*):

    fig = px.bar(df, *x*='File 1', *y*='Similarity', *color*='File 2', *title*='Similarity Bar Chart')

    st.plotly\_chart(fig, *use\_container\_width*=True)

**Histograms**: Shows the distribution of similarity percentages across all file pairs.

*def* plot\_histogram(*df*):

    fig = px.histogram(df, *x*='Similarity', *title*='Similarity Histogram')

    st.plotly\_chart(fig, *use\_container\_width*=True)

**Violin Plots**: Combines summary statistics and distribution data to give a comprehensive view of similarity scores.

*def* plot\_violin(*df*):

    fig = px.violin(df, *y*='Similarity', *x*='File 1', *title*='Similarity Violin Plot')

    st.plotly\_chart(fig, *use\_container\_width*=True)

1. **Code Explanation**

**Core functions**

* **get\_sentences**: Tokenizes text into sentences.

*def* get\_sentences(*text*):

    sentences = tokenize.sent\_tokenize(text)

    return sentences

* **get\_similarity**: Computes cosine similarity between two text inputs.

*def* get\_similarity(*text1*, *text2*):

    text\_list = [text1, text2]

    cv = CountVectorizer()

    count\_matrix = cv.fit\_transform(text\_list)

    similarity = cosine\_similarity(count\_matrix)[0][1]

    return similarity

* **get\_similarity\_list**: Generates pairwise similarity percentages for multiple files.

*def* get\_similarity\_list(*texts*, *filenames*=None):

    similarity\_list = []

    if filenames is None:

        filenames = [*f*"File {i+1}" for i in range(len(texts))]

    for i in range(len(texts)):

        for j in range(i+1, len(texts)):

            similarity = get\_similarity(texts[i], texts[j])

*# Convert similarity to percentage*

            similarity\_percentage = similarity \* 100

            similarity\_list.append((filenames[i], filenames[j], similarity\_percentage))

    return similarity\_list

#### Similarity Calculations

Analyzes word occurrences to determine overlaps between files.

1. **Acknowledgments**

We acknowledge the open-source libraries and tools that made this project possible, including NLTK, Scikit-learn, and Streamlit.