

CSE3086– NoSQL Databases

J Component - Project Report

Review III

PLAGARISM CHECKER

20MIA1136
20MIA1139

Rachana Supriya Nandipati
Rishikesh Reddy Chittedi

MTech CSE with Specialization in Business Analytics

Submitted to

Dr.A. Bhuvaneswari,
Assistant Professor Senior,
SCOPE, VIT, Chennai

School of Computer Science and Engineering



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

November 2023



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Computing Science and Engineering

VIT Chennai

Vandalur - Kelambakkam Road, Chennai - 600 127

FALL SEM 23-24

Worklet details

| | | |
|-----------------------------|--|-------------------|
| Programme | M.Tech with Specialization in Business Analytics | |
| Course Name / Code | NoSQL Databases – CSE3086 | |
| Slot | C2 | |
| Faculty Name | Dr.A. Bhuvaneswari | |
| J Component | Review III | |
| Team Members Name Reg. No | 20MIA1136 | Rachana Nandipati |
| | 20MIA1139 | Rishikesh Reddy |

Team Members(s) Contributions – Tentatively planned for implementation:

| <i>Worklet Tasks</i> | <i>Contributor's Names</i> |
|-------------------------------------|-------------------------------------|
| Dataset Collection | Rishikesh Reddy |
| Preprocessing | Rishikesh Reddy |
| Architecture/ Model/ Flow diagram | Rishikesh Reddy |
| Model building (suitable algorithm) | Rishikesh Reddy & Rachana Nandipati |
| Results – Tables, Graphs | Rachana Nandipati |
| Technical Report writing | Rachana Nandipati |
| Presentation preparation | Rachana Nandipati |

ABSTRACT

Detecting and preventing plagiarism is crucial to maintaining academic integrity and ensuring the originality of content. The Plagiarism Checker Project aims to provide a robust and efficient solution for detecting plagiarism in text documents using MongoDB as the backend database. This project utilizes MongoDB's adaptable and scalable NoSQL database structure for the storage and organization of an array of text documents. MongoDB's document-oriented model allows for efficient retrieval and analysis of textual data, making it an ideal choice for plagiarism detection systems.

The key components of this project include document storage, similarity analysis, and reporting. Users can submit text documents in the user-friendly interface created for analysis, and the system will compare them against a database of previously stored documents to identify potential instances of plagiarism. The system also provides detailed reports, highlighting similarities. The Plagiarism Checker Project by MongoDB presents a comprehensive solution to address the pressing issue of plagiarism. It harnesses MongoDB's flexible and scalable database architecture, sophisticated similarity analysis algorithms, user-friendly interface, and reporting capabilities to offer an efficient and effective plagiarism detection system. This project plays a crucial role in promoting academic integrity and originality while providing valuable feedback to users to improve their content creation practices.

1. INTRODUCTION

The Plagiarism Checker Project is a solution designed to address the growing concern of plagiarism in various domains by leveraging the capabilities of MongoDB, a NoSQL database. Plagiarism detection is essential to maintain integrity and originality in academia, journalism, and content creation.

Problem Statement

Finding the plagiarized parts of the assignments is a very slow work for the professors. Even with a limited number of texts it relies on the teacher's ability to read and remember every submission. As the process of finding plagiarized parts in assignment is based on the teacher's ability to remember all that he or she has read, the results may be incomplete. Some clear cases of copy and paste may easily be overlooked. And since the workload cannot be shared between multiple assistants. Thus, we are introducing system for checking Plagiarism in assignments.

Objectives

- **Efficient Document Storage:** MongoDB's schema-less design allows for seamless storage and retrieval of text documents. This feature ensures scalability, making it well-suited for handling large and diverse collections of textual data efficiently.
- **Similarity Analysis:** The system incorporates advanced algorithms, including cosine similarity and other relevant metrics, to analyze the similarity between documents. This analysis helps identify potential instances of plagiarism, ensuring accurate results.
- **User-Friendly Interface:** The project offers an intuitive user interface that simplifies the process of submitting documents for analysis. Users receive comprehensive reports highlighting similarities and potential sources, making it user-friendly and accessible.
- **Scalability:** MongoDB's horizontal scaling capabilities enable the system to accommodate a growing number of documents and user demands. As the database accumulates data, it can efficiently manage and query large volumes of textual content.
- **Accuracy:** The primary goal of the project is to deliver highly accurate results in plagiarism detection. It aims to minimize both false positives and false negatives, ensuring that instances of plagiarism are identified correctly.

Challenges

- **Data Preprocessing:** Effective data preprocessing is crucial to remove noise from textual data, such as stop words and formatting inconsistencies. This step is essential to ensure accurate similarity analysis.
- **Data Volume and Scalability:** As the database grows with an increasing number of documents, managing and querying the data efficiently becomes a challenge. Ensuring that MongoDB can scale seamlessly is vital to handle the expanding data volume.
- **User Interface Design:** Designing an intuitive and user-friendly interface is essential for users to interact seamlessly with the plagiarism checker. This requires a deep understanding of user needs and workflows to create an interface that enhances the user experience.

The Plagiarism Checker Project, by MongoDB, offers a robust solution to address plagiarism concerns efficiently. It focuses on accurate similarity analysis, scalability, and a user-friendly interface to provide an effective tool for plagiarism detection across various domains.

2. LITERATURE SURVEY

| Sl. no | Title | Author / Journal name / Year | Technique | Result |
|--------|--|--|--|---|
| 1 | An automated system for plagiarism detection using the internet | Allan Knight, Kevin Almeroth, Bruce Bimber Spring Conference, 2009 | Intelligent sentence selection approach | Aiming to reduce query numbers for plagiarism detection. It suggests utilizing pattern recognition techniques to improve identification and further reduce queries. |
| 2 | Detecting Text Similarity on a Scalable No-SQL Database Platform | Sergey Butakov, Stepan Murzintsev, Aleksandr Tskhai International Conference on Platform Technology and Service (PlatCon), 2016 | MongoDB, scalable "key-value" data structure | It achieves a processing speed of 80ms per KB, suitable for similarity detection tasks, with the potential to process 85,000 documents in 24 hours. |

| | | | | |
|---|---|--|--|--|
| 3 | Using Natural Language Processing Techniques and Fuzzy-Semantic Similarity for Plagiarism Detection | Deepa Gupta, Vani K, Charan Kamal Singh International Conference on Advances in Computing, Communications and Informatics, 2014 | NLP techniques, lemmatization, stop word removal, POS tagging, and fuzzy-semantic similarity measures. | The proposed method, POSPIFS, outperforms others in terms of accuracy and efficiency. |
| 4 | High Performance Plagiarism Detection Using Rabin's fingerprint and Adaptive N-gram Methodologies | Karuma Chege, Dr George Okeyo, Dr Richard Rimiru International Journal of Scientific & Engineering Research, 2016 | Rabin's fingerprinting scheme | The research explores Rabin's fingerprinting scheme as an efficient replacement for MD5 and SHA-1, with Rabin's fingerprint. The effective n-gram size for plagiarism detection is 4, and the addition of n-gram rolling improves effectiveness by a factor of 3.02. |

| | | | | |
|---|---|--|--|--|
| 5 | A Plagiarism Detection Technique for Java Program Using Bytecode Analysis | Jeong-Hoon Ji, Gyun Woo, and Hwan-Gue Cho Third International Conference on Convergence and Hybrid Information Technology, 2008 | Detection technique for Java programs using bytecode | Achieved meaningful correlations between bytecode and source code similarities. The technique employs a local alignment algorithm for bytecode comparison. |
| 6 | Using Microsoft SQL Server platform for plagiarism detection | Vladislav Shcherbinin, Sergey Butakov PAN'09 workshop, 2009 | Fingerprinting-based algorithm | The paper presents a plagiarism detection approach using Microsoft SQL Server platform. It employs a fingerprinting-based algorithm and Levenshtein's metric for marking plagiarism. |

| | | | | |
|---|---|---|---|---|
| 7 | Work in Progress: Developing Arabic Plagiarism Detection Tool for E-Learning Systems | Salha M. Alzahrani, Naomie Salim, Mohammed M. Alsofyani International Association of Computer Science and Information Technology - Spring Conference, 2009 | Arabic Plagiarism Detection (APD) Tool | The paper introduces an Arabic Plagiarism Detection (APD) Tool for e-learning systems, enabling automatic detection and highlighting of plagiarism in Arabic documents. |
| 8 | An IR-based Approach Utilising Query Expansion for Plagiarism Detection in MEDLINE | Rao Muhammad Adeel Nawab, Mark Stevenson and Paul Clough Journal of computational biology and bioinformatics, april 2015 | Information Retrieval (IR)-based approach | (IR) based approach for plagiarism detection, using query expansion. The proposed method outperforms Kullback-Leibler Distance. |

| | | | | |
|----|--|--|-----------------|--|
| 9 | Automatic Generation of Plagiarism Detection Among Student Programs | <p>Edita Roxas, Nathalie Rose Lim and Natasja Bautista</p> <p>7th International Conference on Information Technology Based Higher Education and Training, 2016</p> | Transformations | <p>The paper presents a plagiarism detection system for programming languages using transformations, providing flexibility and accuracy. Future work includes handling additional programming paradigms and further transformations.</p> |
| 10 | Reliable plagiarism detection system based on deep learning approaches | <p>Mohamed A. El-Rashidy, Ramy G. Mohamed, Nawal A. El-Fishawy¹, Marwa A. Shouman¹</p> <p>Springer, 2022</p> | LSTM | <p>An intelligent deep learning system for text plagiarism detection using a new database with 42 features reflecting various text similarities.</p> |

3. DATASET AND DATABASE SPECIFIC TOOL TO BE USED (DETAILS)

A plagiarism checker is a tool used to detect similarities between documents, and in your case, you want to use text files as the dataset for plagiarism detection.

1. Data Collection and Storage:

- ✓ Begin by gathering the text documents that need to be checked for plagiarism. These documents can be in the form of text files, each representing a piece of content authored by students or any other source.
- ✓ Next, set up a MongoDB database to store these documents. MongoDB is a NoSQL database that can efficiently store and retrieve text data. You'll create collections within the database to organize your documents.

2. MongoDB Database Setup:

- ✓ Create a new MongoDB database to hold your plagiarism checking data.
- ✓ Within this database, create one or more collections, each representing a group of documents. For instance, you could have a "student_documents" collection to store student essays, papers, or assignments.

3. Importing Data:

- ✓ To populate your MongoDB collection(s), you can write a script or use a MongoDB client tool to insert the text data from your text files into the appropriate collections. Each document in the collection should include a unique identifier (e.g., a document ID) and the text content itself.

4. Integration with Python:

- ✓ Use the PyMongo library to connect your Python application to the MongoDB database. PyMongo provides a Pythonic way to interact with MongoDB.
- ✓ Retrieve the documents from the MongoDB collection(s) as needed for plagiarism checking.

The text content retrieved from MongoDB will be used as the input for your plagiarism detection algorithm. This way we can set up a plagiarism checker that integrates with MongoDB to efficiently manage and analyze a dataset of text documents for plagiarism detection. This system will allow you to maintain a secure and organized repository of documents while performing plagiarism checks as needed.

4. ALGORITHMS / TECHNIQUES DESCRIPTION

Pseudocode

1. Import necessary libraries and modules
2. Define MongoDB connection settings (URI, database name, collection name)
3. Initialize MongoDB client and select the database and collection
4. Attempt to establish a connection to MongoDB
5. Initialize the MongoDB client and select the database and collection
6. Print an error message and exit the program if there's a connection issue
7. Define a function to perform text vectorization using TF-IDF
8. Use scikit-learn's TfidfVectorizer to convert text into TF-IDF vectors
9. Return the TF-IDF vectors as NumPy arrays
10. Define a function to calculate cosine similarity between two documents
11. Use scikit-learn's cosine_similarity to calculate cosine similarity
12. Return the similarity score as a float
13. Initialize an empty dictionary to store plagiarism results
14. Attempt to retrieve student documents from the MongoDB collection
15. Use the collection.find() method to query MongoDB and retrieve student documents
16. Store the retrieved documents in a list
17. Vectorize the text content of student documents
18. Use the vectorize () function to convert text into TF-IDF vectors
19. Iterate through pairs of student documents
20. Avoid comparing a document with itself
21. Calculate the similarity score between the TF-IDF vectors of two documents

22. Use the similarity () function to calculate cosine similarity
23. Convert the plagiarism_results dictionary to a JSON-formatted string
24. Return the JSON string containing the plagiarism detection results
25. If the script is executed directly (not imported), call the check_plagiarism function
26. Print the results

Techniques used

The plagiarism checker will be using Cosine similarity algorithm to find the similarity between the documents from the extracted keywords, and etc.

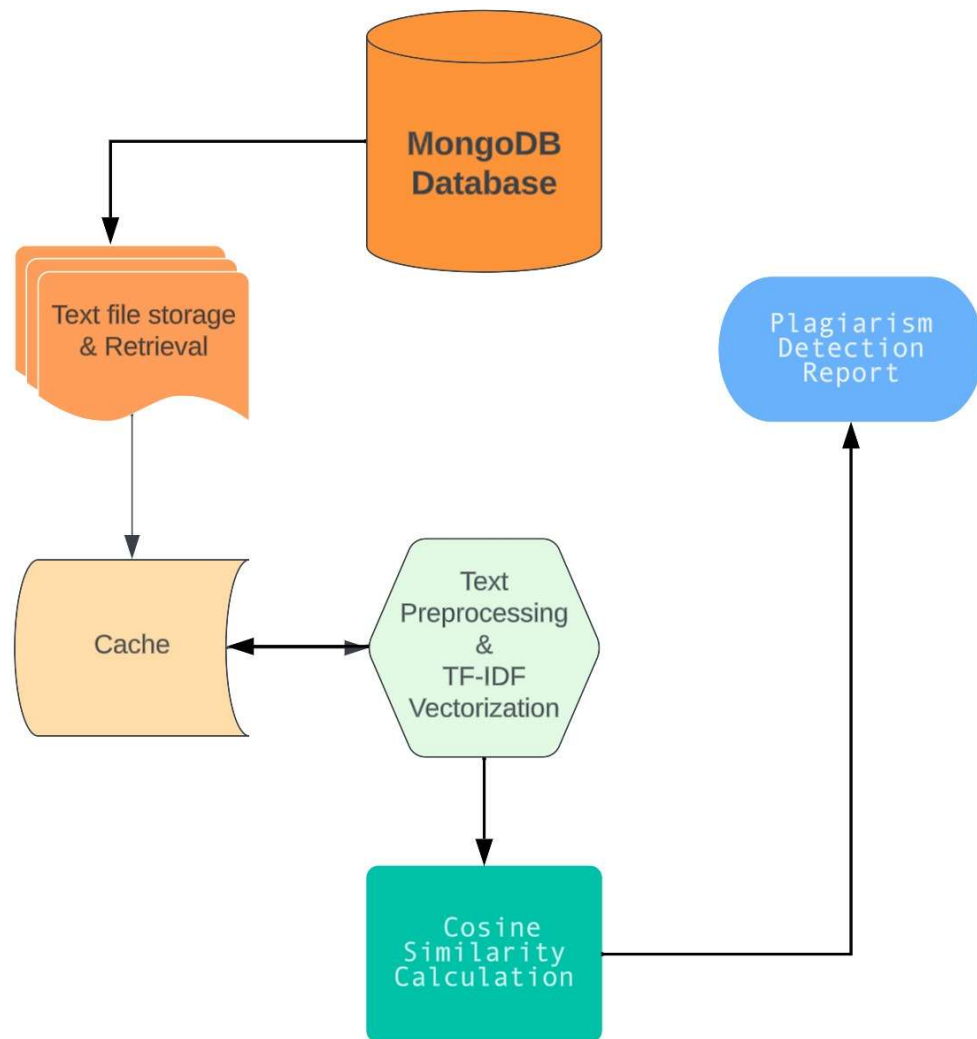
Cosine similarity measures the similarity between two vectors of an inner product space. It is measured by the cosine of the angle between two vectors and determines whether two vectors are pointing in roughly the same direction. Let x and y be two vectors for comparison. Using the cosine measure as a similarity function, we have

$$\text{sim}(x,y) = (x \cdot y) / (\|x\| \|y\|)$$

where $\|x\|$ is the Euclidean norm of vector $x=(x_1,x_2,\dots,x_p)$, defined as $x_1^2+x_2^2+\dots+x_p^2$. Conceptually, it is the length of the vector. Similarly, $\|y\|$ is the Euclidean norm of vector y . The measure computes the cosine of the angle between vectors x and y . A cosine value of 0 means that the two vectors are at 90 degrees to each other (orthogonal) and have no match. The closer the cosine value to 1, the smaller the angle and the greater the match between vectors.

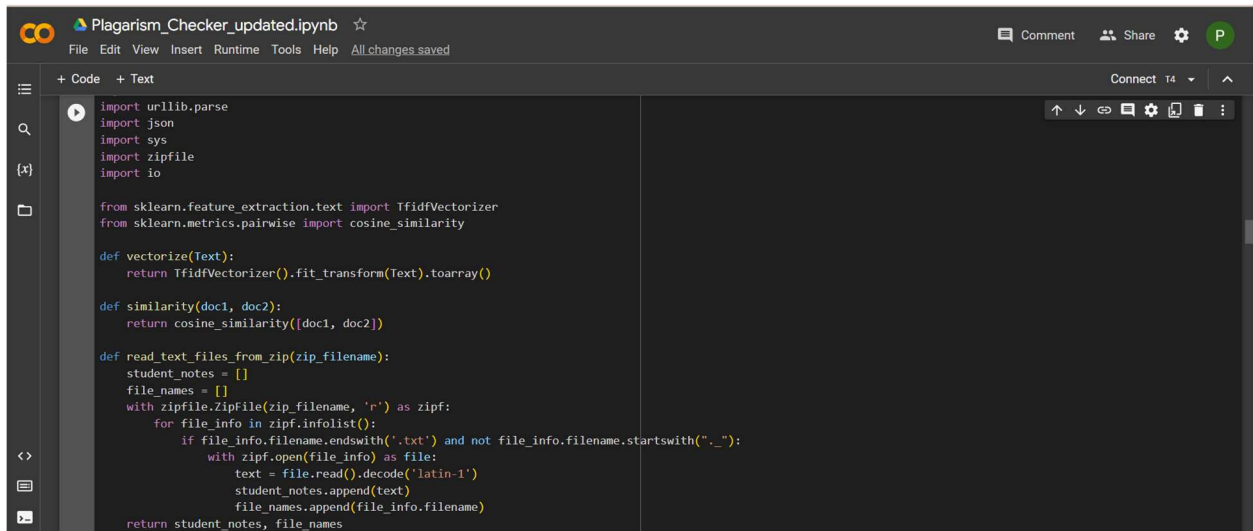
TF-IDF Vectorization (TfidfVectorizer): The program uses the TF-IDF (Term Frequency-Inverse Document Frequency) vectorization technique to convert the text data into numerical feature vectors. This technique assigns weights to words in a document based on their frequency and importance relative to the entire corpus of documents. The TfidfVectorizer from scikit-learn is used for this purpose.

5. ARCHITECTURE DIAGRAM OF THE PROPOSED WORK / SYSTEM DESIGN



6. IMPLEMENTATION

- Retrieve all files
- Get the content for each file
- Check cosine similarity value among each and every file contents
- Conversion of dictionary into JSON format using API
- Display the results



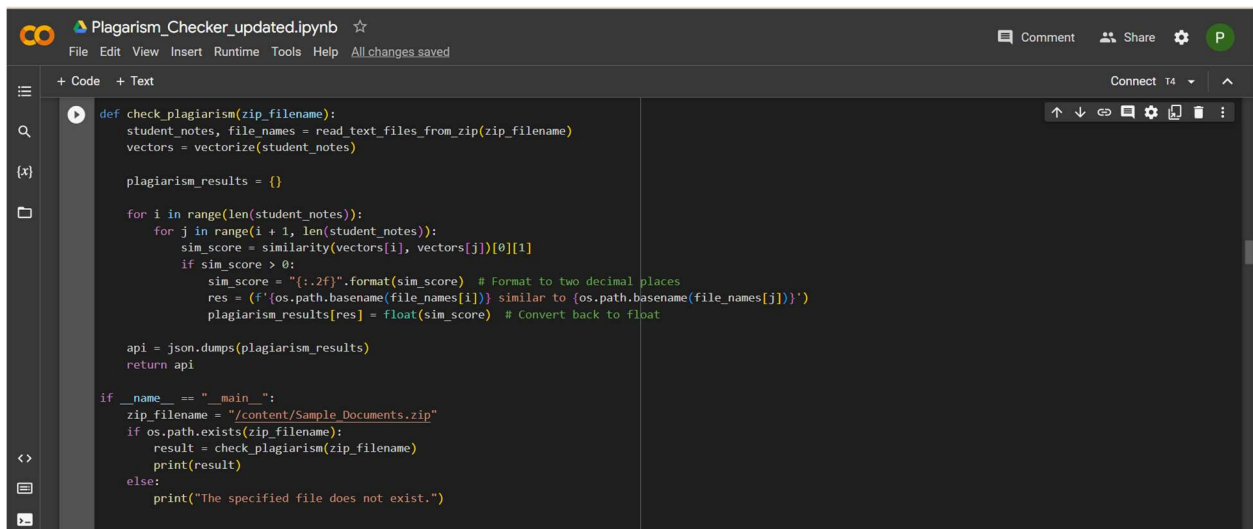
```
Plagarism_Checker_updated.ipynb ☆
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
import urllib.parse
import json
import sys
import zipfile
import io

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

def vectorize(Text):
    return TfidfVectorizer().fit_transform(Text).toarray()

def similarity(doc1, doc2):
    return cosine_similarity([doc1, doc2])

def read_text_files_from_zip(zip_filename):
    student_notes = []
    file_names = []
    with zipfile.ZipFile(zip_filename, 'r') as zipf:
        for file_info in zipf.infolist():
            if file_info.filename.endswith('.txt') and not file_info.filename.startswith("__"):
                with zipf.open(file_info) as file:
                    text = file.read().decode('latin-1')
                    student_notes.append(text)
                    file_names.append(file_info.filename)
    return student_notes, file_names
```



```
Plagarism_Checker_updated.ipynb ☆
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
def check_plagiarism(zip_filename):
    student_notes, file_names = read_text_files_from_zip(zip_filename)
    vectors = vectorize(student_notes)

    plagiarism_results = {}

    for i in range(len(student_notes)):
        for j in range(i + 1, len(student_notes)):
            sim_score = similarity(vectors[i], vectors[j])[0][1]
            if sim_score > 0:
                sim_score = "{:.2f}".format(sim_score) # Format to two decimal places
                res = ('{os.path.basename(file_names[i])} similar to {os.path.basename(file_names[j])}')
                plagiarism_results[res] = float(sim_score) # Convert back to float

    api = json.dumps(plagiarism_results)
    return api

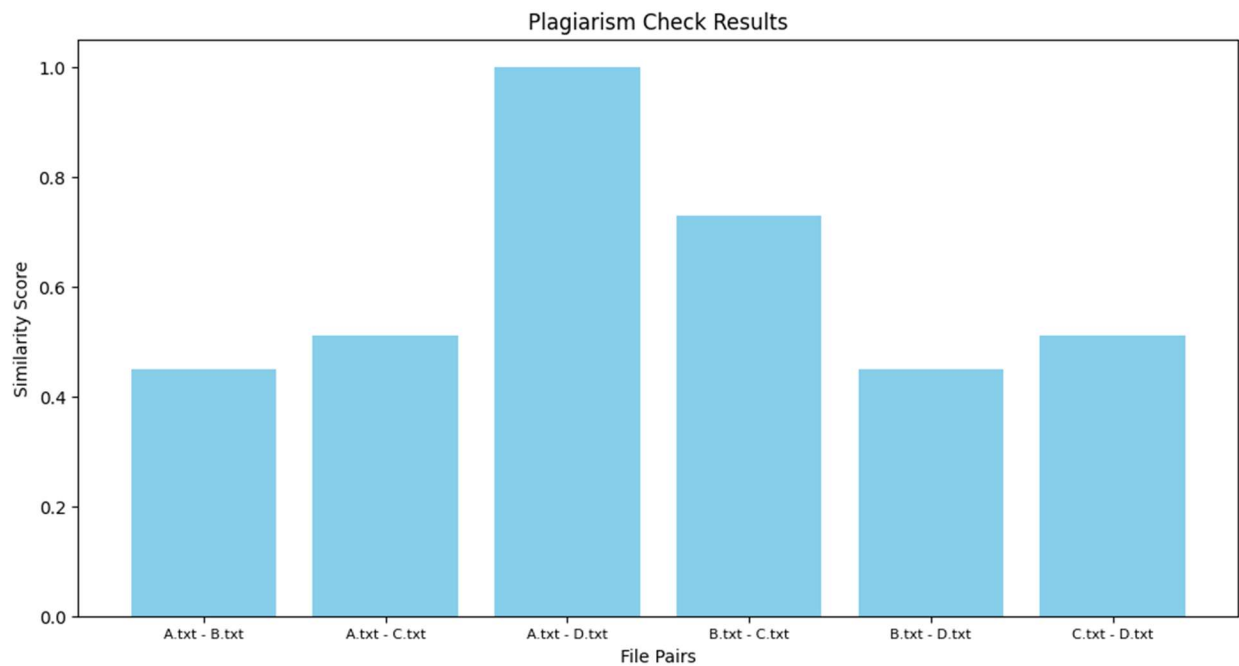
if __name__ == "__main__":
    zip_filename = "/content/Sample Documents.zip"
    if os.path.exists(zip_filename):
        result = check_plagiarism(zip_filename)
        print(result)
    else:
        print("The specified file does not exist.")
```

SAMPLE OUTPUT

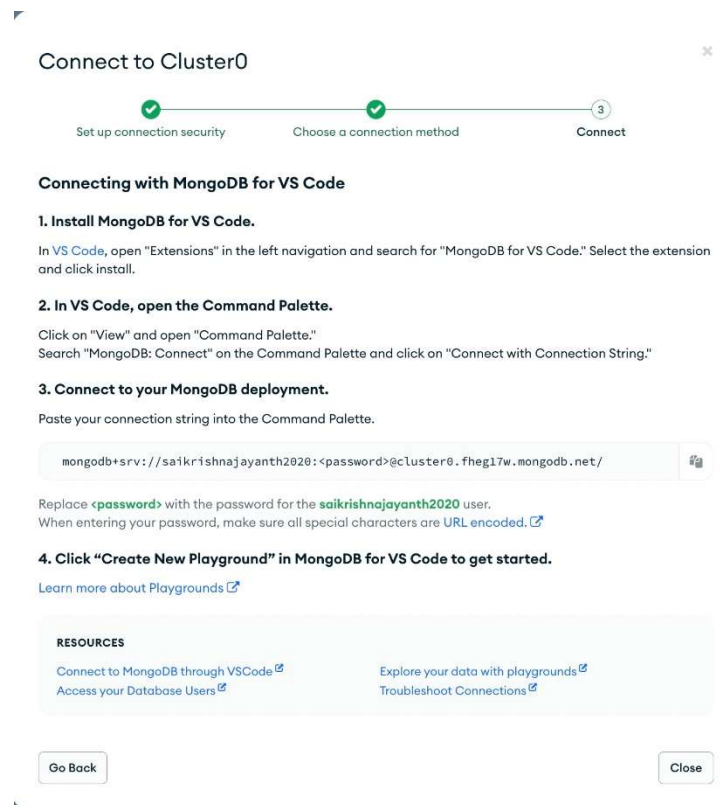
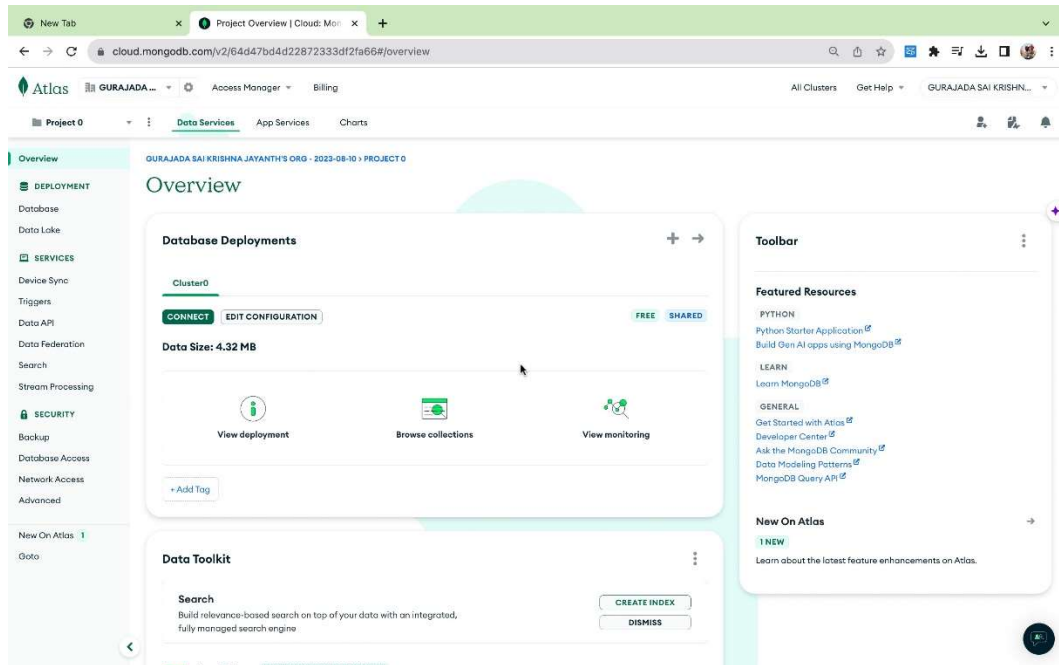
```
➡ {"A.txt similar to B.txt": 0.45, "A.txt similar to C.txt": 0.51, "A.txt similar to D.txt": 1.0, "B.txt similar to C.txt": 0.73, "B.txt similar to D.txt": 0.45, "C.txt si
```

```
▶ plt.figure(figsize=(12, 6))
plt.bar(x, similarity_scores, color='skyblue')
plt.xticks(x, [f'{file1} - {file2}' for file1, file2 in zip(file1_names, file2_names)], rotation=0) # Horizontal labels
plt.xlabel('File Pairs')
plt.ylabel('Similarity Score')
plt.title('Plagiarism Check Results')
plt.gca().xaxis.set_major_locator(plt.MultipleLocator(1)) # Set the x-axis tick spacing
plt.xticks(fontsize=8) # Adjust font size for labels

plt.show()
else:
    print("No plagiarism found.")
else:
    print("The specified file does not exist.")
```



MONGODB INTEGRATION



CODE: app.py

```
from flask import Flask, render_template, request, redirect, url_for

from pymongo import MongoClient

from werkzeug.utils import secure_filename

import os

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.metrics.pairwise import cosine_similarity

app = Flask(__name__)

# MongoDB Configuration

mongodb_url = "mongodb+srv://rishikeshreddy:loki1234@cluster0.w1uhnmx.mongodb.net/"

client = MongoClient(mongodb_url)

db = client['plagarism_checker']

collection = db['documents']

# Configuring upload folder

app.config['UPLOAD_FOLDER'] = 'uploads'

if not os.path.exists(app.config['UPLOAD_FOLDER']):

    os.makedirs(app.config['UPLOAD_FOLDER'])

def vectorize(Text):

    return TfidfVectorizer().fit_transform(Text).toarray()

def similarity(doc1, doc2):

    return cosine_similarity([doc1, doc2])

def read_documents_from_mongodb():
```

```

documents = list(collection.find())

return [doc["content"] for doc in documents]

def check_plagiarism(doc1, doc2):

    vectors = vectorize([doc1, doc2])

    sim_score = similarity(vectors[0], vectors[1])[0][1]

    return round(sim_score, 2)

@app.route('/')

def home():

    documents = collection.find()

    return render_template('home.html', documents=documents)

@app.route('/upload', methods=['POST'])

def upload_file():

    if 'file' not in request.files:

        return redirect(request.url)

    file = request.files['file']

    if file.filename == "":

        return redirect(request.url)

    if file:

        filename = secure_filename(file.filename)

        file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))

        with open(os.path.join(app.config['UPLOAD_FOLDER'], filename), 'r') as file_content:

            content = file_content.read()

        document = {

```

```

        'filename': filename,

        'path': os.path.join(app.config['UPLOAD_FOLDER'], filename),

        'content': content

    }

    collection.insert_one(document)

    return redirect(url_for('home'))

@app.route('/check_plagiarism')
def check_plagiarism_route():

    # Assuming there are two documents in the collection

    documents = list(collection.find(limit=2))

    if len(documents) == 2:

        doc1_content = documents[0]["content"]

        doc2_content = documents[1]["content"]

        similarity_score = check_plagiarism(doc1_content, doc2_content)

        return f"Similarity Score for these two documents: {similarity_score}"

    else:

        return "Please upload two documents before checking plagiarism."

if __name__ == '__main__':

    app.run(debug=True, port=5000)

```

```
Get Started  app.py x  styles.css  home.html  plagiarism_results.html
1 from flask import Flask, render_template, request, redirect, url_for
2 from pymongo import MongoClient
3 from werkzeug.utils import secure_filename
4 import os
5 from sklearn.feature_extraction.text import TfidfVectorizer
6 from sklearn.metrics.pairwise import cosine_similarity
7
8 app = Flask(__name__)
9
10 # MongoDB Configuration
11 mongodb_url = "mongodb+srv://saikrishnajayanth2020:jayanth@cluster0.fheg17w.mongodb.net/"
12 client = MongoClient(mongodb_url)
13 db = client["plagiarism_checker"]
14 collection = db["documents"]
15
16 # Configuring upload folder
17 app.config['UPLOAD_FOLDER'] = 'uploads'
18 if not os.path.exists(app.config['UPLOAD_FOLDER']):
19     os.makedirs(app.config['UPLOAD_FOLDER'])
20
21 def vectorize(Text):
22     return TfidfVectorizer().fit_transform(Text).toarray()
23
24 def similarity(doc1, doc2):
25     return cosine_similarity([doc1, doc2])
26
27 def read_documents_from_mongodb():
28     documents = list(collection.find())
29     return [doc["content"] for doc in documents]
30
31 def check_plagiarism(doc1, doc2):
32     vectors = vectorize([doc1, doc2])
33     sim_score = similarity(vectors[0], vectors[1])[0][1]
34     return round(sim_score, 2)
35
36 @app.route('/')
37 def home():
38     documents = collection.find()
39     return render_template('home.html', documents=documents)
40
41 @app.route('/upload', methods=['POST'])
42 def upload_file():
43     if 'file' not in request.files:
44         return redirect(request.url)
45     file = request.files['file']
```

CODE: home.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">

<style>

body {

font-family: 'Arial', sans-serif;

background-color: #f4f4f4;

```
margin: 0;

padding: 0;
}

.container {

    max-width: 800px;

    margin: 50px auto;

    background-color: #fff;

    padding: 20px;

    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

    border-radius: 8px;

    text-align: center;
}

h1 {

    color: #333;
}

form {

    margin-top: 20px;
}

input[type="file"] {

    padding: 10px;

    margin-bottom: 10px;

    border: 1px solid #ddd;

    border-radius: 4px;
```

```
width: 100%;

box-sizing: border-box;

}

button {

background-color: #4caf50;

color: white;

padding: 10px 20px;

border: none;

border-radius: 4px;

cursor: pointer;

}

button:hover {

background-color: #45a049;

}

.check-btn {

display: inline-block;

margin-top: 20px;

padding: 10px 20px;

background-color: #007bff;

color: white;

text-decoration: none;

border-radius: 4px;

}
```

```
.check-btn:hover {  
    background-color: #0056b3;  
}  
  
ul {  
    list-style: none;  
    padding: 0;  
    margin: 20px 0;  
}  
  
li {  
    background-color: #eee;  
    padding: 10px;  
    margin: 5px 0;  
    border-radius: 4px;  
}  
  
</style>  
  
<title>Plagiarism Checker</title>  
  
</head>  
  
<body>  
  
    <div class="container">  
  
          
  
        <h1>Plagiarism Checker</h1>  
  
        <form action="/upload" method="post" enctype="multipart/form-data">  
  
            <input type="file" name="file" accept=".txt" required>
```

```

        <button type="submit">Upload Document</button>

    </form>

    <a href="{{ url_for('check_plagiarism_route') }}" class="check-btn">Check Plagiarism</a>

</ul>

    {% for document in documents %}

        <li>{{ document['filename'] }}</li>

    {% endfor %}

</ul>

</div>

</body>

</html>

```

The screenshot shows a code editor with the following content:

```

templates > home.html
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta http-equiv="X-UA-Compatible" content="IE=edge">
7      <meta name="viewport" content="width=device-width, initial-scale=1.0">
8      <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
9      <title>Plagiarism Checker</title>
10 </head>
11
12 <body>
13     <div class="container">
14         <h1>Plagiarism Checker</h1>
15
16         <form action="/upload" method="post" enctype="multipart/form-data">
17             <input type="file" name="file" accept=".txt" required>
18             <button type="submit">Upload Document</button>
19         </form>
20
21         <a href="{{ url_for('check_plagiarism_route') }}" class="check-btn">Check Plagiarism</a>
22
23         <ul>
24             {% for document in documents %}
25                 <li>{{ document['filename'] }}</li>
26             {% endfor %}
27         </ul>
28     </div>
29 </body>
30 </html>
31
32

```

The status bar at the bottom indicates: Ln 32, Col 1 | Spaces: 4 | UTF-8 | LF | Django HTML

CODE: styles.css

```
body {  
  
    font-family: Arial, sans-serif;  
  
    margin: 0;  
  
    padding: 0;  
  
    background-color: #3498db;  
  
    color: #ecf0f1;  
  
}  
  
.container {  
  
    width: 80%;  
  
    margin: 0 auto;  
  
    text-align: center;  
  
}  
  
h1 {  
  
    color: #ecf0f1;  
  
}  
  
form {  
  
    margin-top: 20px;  
  
}  
  
input {  
  
    padding: 10px;
```

```
}
```

```
button {
```

```
    background-color: #2ecc71;
```

```
    color: white;
```

```
    padding: 10px 20px;
```

```
    border: none;
```

```
    border-radius: 5px;
```

```
    cursor: pointer;
```

```
}
```

```
button:hover {
```

```
    background-color: #27ae60;
```

```
}
```

```
.check-btn {
```

```
    display: inline-block;
```

```
    background-color: #3498db;
```

```
    color: white;
```

```
    padding: 15px 20px;
```

```
    text-decoration: none;
```

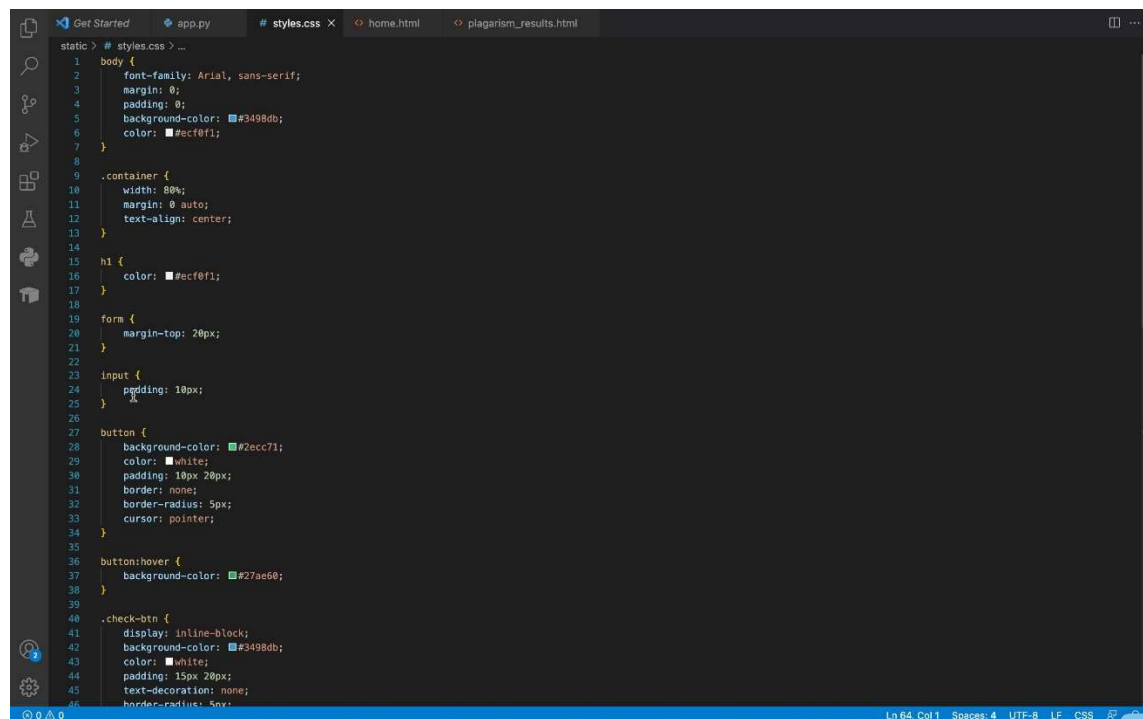
```
    border-radius: 5px;
```

```
    transition: background-color 0.3s;
```

```
    margin-top: 10px;
```

```
}
```

```
.check-btn:hover {  
  
    background-color: #2980b9;  
  
}  
  
ul {  
  
    list-style-type: none;  
  
    padding: 0;  
  
    margin-top: 20px;  
  
}  
  
li {  
  
    margin-bottom: 10px;  
  
}
```



CODE: plagiarism_results.html

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Plagiarism Check Results</title>

    <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='styles.css') }}">

    <style>

        body {

            font-family: 'Arial', sans-serif;

            background-color: #f8fafa;

            margin: 0;

            padding: 0;

            text-align: center;

        }

        h1 {

            color: #007bff;

            margin-top: 50px;

        }

        ul {

            list-style: none;
```

```
padding: 0;

margin-top: 20px;

}

li {

background-color: #fff;

padding: 15px;

margin: 10px 0;

border-radius: 8px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

}

p {

color: #28a745;

margin-top: 20px;

}

.no-plagiarism {

color: #dc3545;

margin-top: 20px;

}

</style>

</head>

<body>

<h1>Plagiarism Check Results</h1>
```

```

{% if similarity_scores %}

<ul>

    {% for score in similarity_scores %}

        <li>Similarity Score between Doc {{ score[0] + 1 }} and Doc {{ score[1] + 1 }}: {{ score[2]
        }}</li>

    {% endfor %}

</ul>

{% else %}

<p class="no-plagiarism">No plagiarism found.</p>

{% endif %}

</body>

</html>

```

The screenshot shows a code editor with the following content in the 'plagiarism_results.html' file:

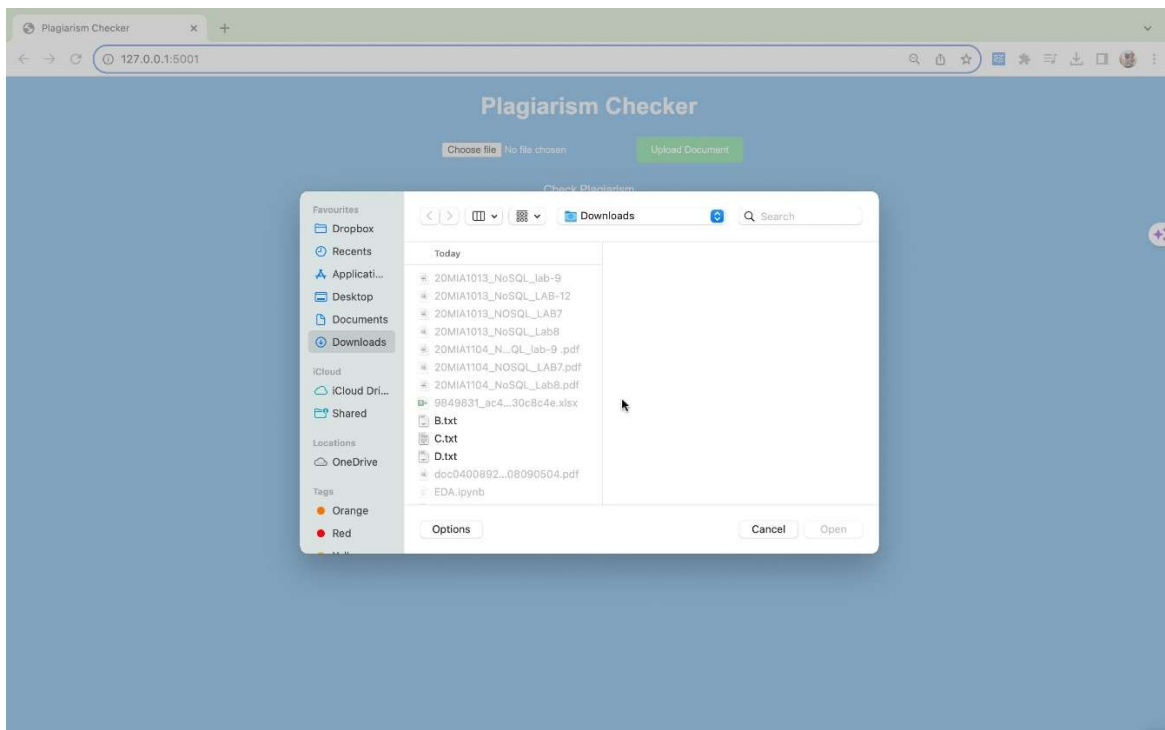
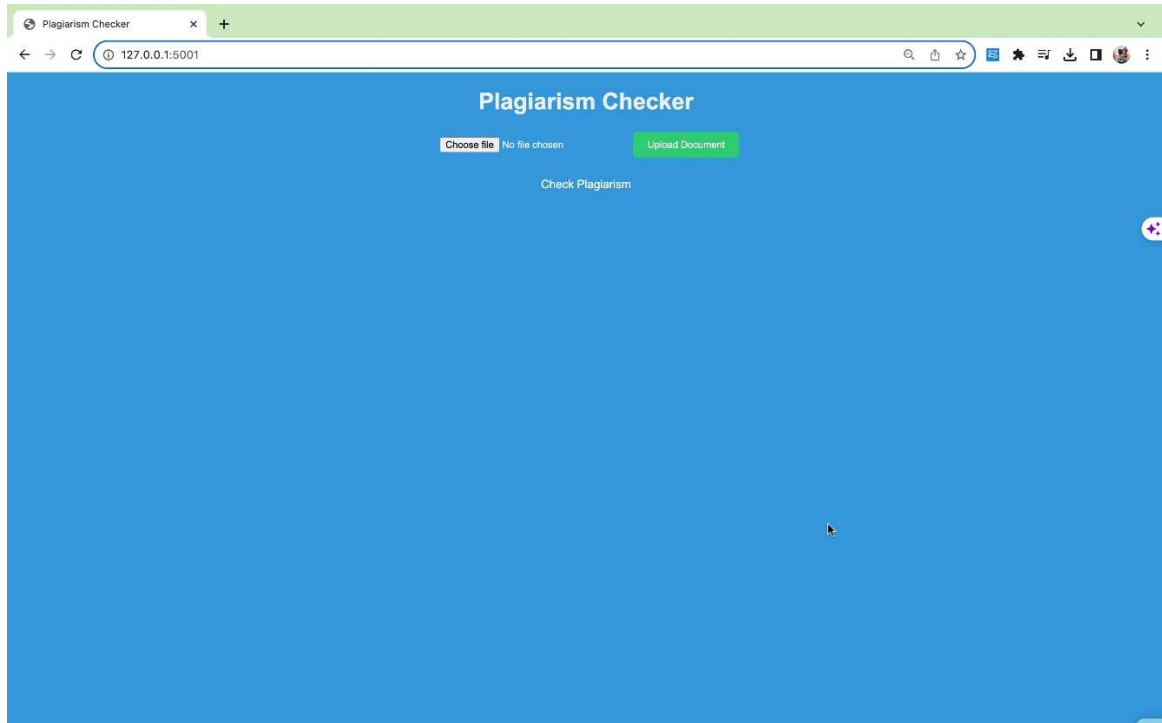
```

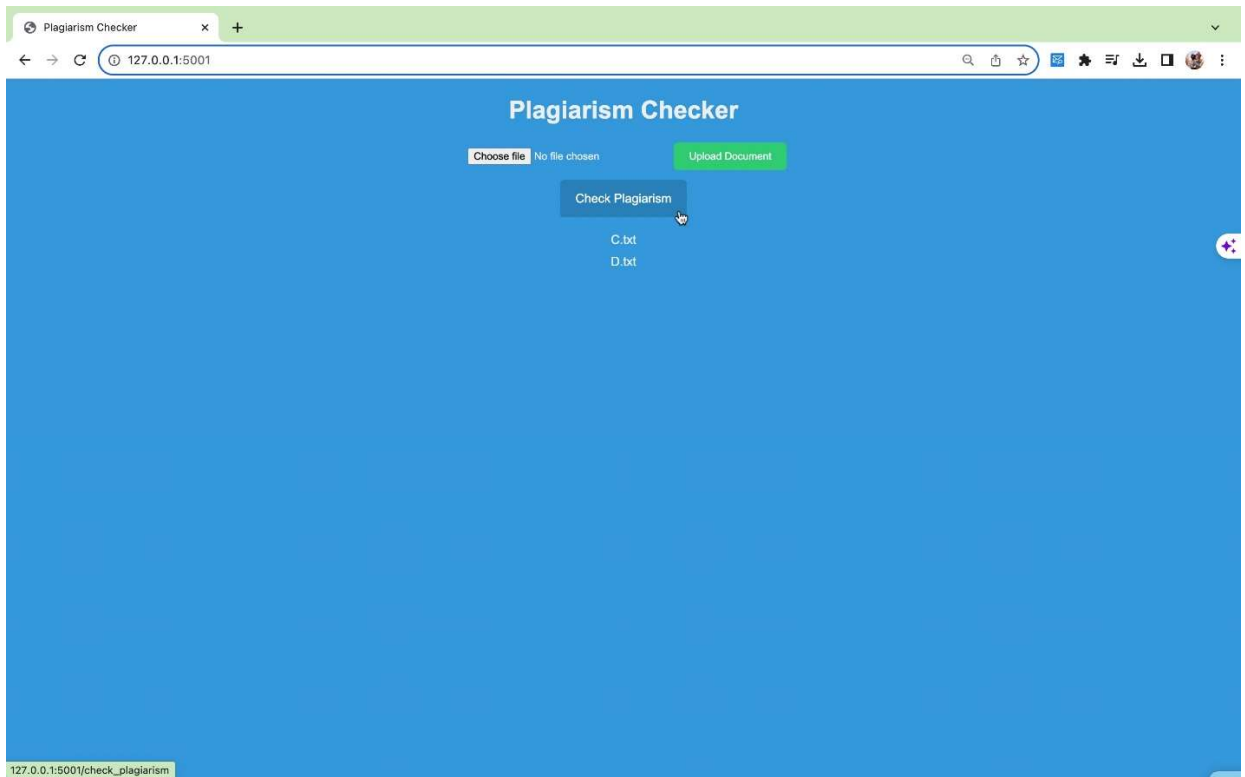
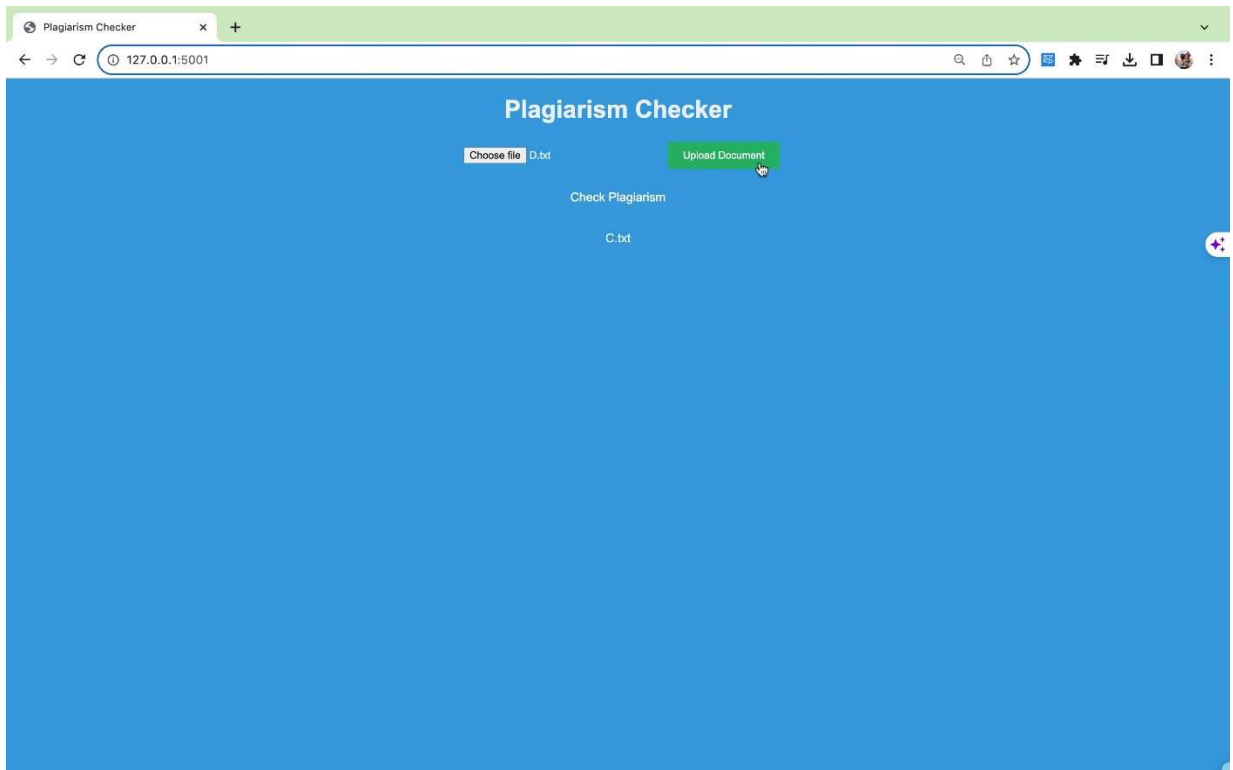
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Plagiarism Check Results</title>
8      <link rel="stylesheet" type="text/css" href="{% url_for 'static', filename='styles.css' %}">
9  </head>
10 <body>
11     <h1>Plagiarism Check Results</h1>
12
13     {% if similarity_scores %}
14         <ul>
15             {% for score in similarity_scores %}
16                 <li>Similarity Score between Doc {{ score[0] + 1 }} and Doc {{ score[1] + 1 }}: {{ score[2] }}</li>
17             {% endfor %}
18         </ul>
19     {% else %}
20         <p>No plagiarism found.</p>
21     {% endif %}
22 </body>
23 </html>
24

```

The editor interface includes a sidebar with icons for file explorer, search, and other tools. The status bar at the bottom indicates the current line and column (Ln 24, Col 1), the number of spaces (4), the encoding (UTF-8), the line ending (LF), and the file type (Django HTML).

7. RESULTS AND DISCUSSION





cloud.mongodb.com/v2/64d47bd4d22872333df2fa66#/metrics/replicaSet/64d47c262fe6026f92beb261/explorer/plagiarism_checker/document...

Atlas GURAJADA ... Access Manager Billing All Clusters Get Help GURAJADA SAI KRISHN...

Project 0 Data Services App Services Charts

Overview

DEPLOYMENT

Database

Data Lake

SERVICES

Device Sync

Triggers

Data API

Data Federation

Search

Stream Processing

SECURITY

Backup

Database Access

Network Access

Advanced

Quto

+ Create Database

Search Namespaces

plagiarism_checker.documents

STORAGE SIZE: 48KB LOGICAL DATA SIZE: 16.79KB TOTAL DOCUMENTS: 2 INDEXES TOTAL SIZE: 36KB

Find Indexes Schema Anti-Patterns Aggregation Search Indexes

Filter Type a query: { field: 'value' } Reset Apply Options

QUERY RESULTS: 1-2 OF 2

```
{
  "_id": ObjectId("655e37ae245e9abc8c1ab7ab"),
  "filename": "C.txt",
  "path": "uploads/C.txt",
  "content": "ABSTRACT

Smartphones influence one's daily life, as they enhance ente..."
}
```

```
{
  "_id": ObjectId("655e37d3245e9abc8c1ab7ac"),
  "filename": "D.txt",
  "path": "uploads/D.txt",
  "content": "In this paper we measure and quantify how consumer's choice o..."
}
```

System Status: All Good

©2023 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales

127.0.0.1:5001/check_plagiari x +

← → ↺ ⓘ 127.0.0.1:5001/check_plagiarism

Similarity Score for these two documents: 0.56

User Interface

The application utilizes Flask, a web framework for Python, to establish a straightforward user interface. The primary route ('/') renders the 'home.html' template, which displays a list of uploaded documents stored within the MongoDB database.

File Upload

The route '/upload' manages file uploading. When a user submits a file through the HTML form, the file is stored in the 'uploads' directory. The file's metadata, including filename, path, and content, are preserved in a MongoDB document.

Plagiarism Checking

The route '/check_plagiarism' contrasts the content of the two most recently uploaded documents. It fetches the documents from MongoDB, extracts the content, converts them into TF-IDF vectors, and calculates the cosine similarity between the vectors. The similarity score is rounded to two decimal places and presented to the user.

File Handling

The application employs the 'werkzeug.utils.secure_filename' function to sanitize filenames before saving them to the 'uploads' directory. This measure safeguards against potential security vulnerabilities associated with user-uploaded files.

MongoDB Integration

The application leverages the PyMongo library to establish a connection with a MongoDB database. It stores document metadata and file content within a collection named 'documents'.

Overall Workflow

1. The user uploads a document using the HTML form on the '/upload' page.
2. The file is stored in the 'uploads' directory, and its metadata is preserved in a MongoDB document.
3. The user navigates to the '/check_plagiarism' page.
4. The application retrieves the content of the two most recently uploaded documents from MongoDB.

5. The application converts the document content to TF-IDF vectors and calculates the cosine similarity between the vectors.
6. The similarity score is presented to the user.

8. CONCLUSION

The implemented system demonstrates a web application that utilizes Flask, a Python web framework, to create a user interface for uploading documents and checking plagiarism using cosine similarity. MongoDB, a NoSQL database, serves as the backend storage for document metadata and file content.

The application's workflow encompasses file uploading, plagiarism checking, and seamless integration with MongoDB. Users can upload documents through the web interface, and the application stores the file metadata and content in MongoDB. The plagiarism checking feature compares the content of the two most recently uploaded documents, converting them to TF-IDF vectors and calculating the cosine similarity to determine the plagiarism score.

The system effectively utilizes Flask for user interaction, MongoDB for data persistence, and cosine similarity for plagiarism detection. It demonstrates a practical implementation of plagiarism checking using a web application and MongoDB.

9. GITHUB REPOSITORY LINK

https://github.com/RachanaNandipati/Plagiarism_Checker

10. CITATIONS

- [1] Knight, Allan & Almeroth, Kevin & Bimber, Bruce. (2004). An automated system for plagiarism detection using the internet.
- [2] S. Butakov, S. Murzintsev and A. Tskhai, "Detecting Text Similarity on a Scalable No-SQL Database Platform," 2016 International Conference on Platform Technology and Service (PlatCon), Jeju, Korea (South), 2016, pp. 1-5
- [3] D. Gupta, K. Vani and C. K. Singh, "Using Natural Language Processing techniques and fuzzy-semantic similarity for automatic external plagiarism detection," 2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Delhi, India, 2014, pp. 2694-2699
- [4] High Performance Plagiarism Detection Using Rabin's fingerprint and Adaptive N-gram Methodologies Karuma Chege, Dr George Okeyo PhD, Dr Richard Rimiru PhD.
- [5] J. -H. Ji, G. Woo and H. -G. Cho, "A Plagiarism Detection Technique for Java Program Using Bytecode Analysis," 2008 Third International Conference on Convergence and Hybrid Information Technology, Busan, Korea (South), 2008.
- [6] Shcherbinin, Vladislav and Sergey Butakov. "Using Microsoft SQL Server platform for plagiarism detection." (2009).
- [7] S. M. Alzahrani, N. Salim and M. M. Alsofyani, "Work in Progress: Developing Arabic Plagiarism Detection Tool for E-Learning Systems," 2009 International Association of Computer Science and Information Technology - Spring Conference, Singapore, 2009, pp. 105-109
- [8] R. M. A. Nawab, M. Stevenson and P. Clough, "An IR-Based Approach Utilizing Query Expansion for Plagiarism Detection in MEDLINE," in IEEE/ACM Transactions on Computational Biology and Bioinformatics, vol. 14, no. 4, pp. 796-804, 1 July-Aug. 2017,
- [9] R. E. Roxas, N. R. Lim and N. Bautista, "Automatic Generation of Plagiarism Detection Among Student Programs," 2006 7th International Conference on Information Technology Based Higher Education and Training, Ultimo, Australia, 2006, pp. 226-235