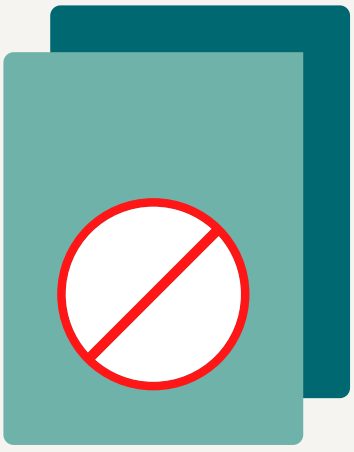
Computer Science Department-UQU
قسم علوم الحاسب الآلي

جامعة أم القرى
UMM AL-QURA UNIVERSITY

TSC

TEXT SIMILARITY CHECKER

NATURAL LANGUAGE PROCESSING PROJECT REPORT



SUPERVISED BY

Dr.Amal Alshmrani

TABLE OF CONTENTS

1 Introduction

1.1 background

A computer program's capacity to comprehend natural language, or human language as it is spoken and written, is known as natural language processing (NLP). It is a part of machine intelligence (AI). NLP has been around for more than 50 years and has linguistics roots. It has numerous practical uses in a range of industries, including corporate intelligence, search engines, and medical research. Computers can now comprehend natural language just like people do thanks to NLP. Natural language processing use artificial intelligence to take real-world input, process it, and make sense of it in a way that a computer can comprehend, regardless of whether the language is spoken or written.

Computers have reading programs and microphones to collect audio, much as people have various sensors like ears to hear and eyes to see. Computers have a program to process their various inputs, just as humans have a brain to do so. The input is eventually translated into computer-readable code during processing.

The creation of algorithms and data preprocessing are the two fundamental stages of natural language processing. Preparing and "cleaning" text data so that computers can analyze it is known as data preprocessing. Preprocessing prepares data for use and highlights text features that an algorithm can use.

1.2 Purpose and Goals

- Build an online platform to check the Similarity of graduation project within the abstract that have been written in graduation project to make you check similarity relative to the files in the system, known as an "Offline checker".

2. Text Similarity

2.1 Plagiarism

The act of presenting someone else's words, ideas, or expressions as one's own unique work is known as plagiarism. The definition of plagiarism in educational contexts varies depending on the institution.

It is regarded as a breach of journalistic ethics and a violation of academic integrity, which is defined as truth and knowledge through intellectual and personal honesty in learning, teaching, and research, fairness, respect, and responsibility. It is punishable by suspension, expulsion from school or the workplace, significant fines, and even imprisonment.

Plagiarism is considered academic dishonesty by educational institutions worldwide. Plagiarism is a perennial problem that affects not only schools and teachers, but also entire societies. Consequently, teachers develop and implement special classroom management methods and implement disciplinary measures to deal with this problem. Despite these efforts, cheating continues due to the fact that students have access to innumerable resources online. It is nevertheless possible for teachers to determine the originality of students' work using a plagiarism checker, which can benefit them, the institution and the education process in general.

Your research papers, blogs, assignments, and websites can be checked for plagiarism with a plagiarism checker.

2. Text Similarity

What Are Some Examples of Plagiarism?

1. Complete Plagiarism:

Complete plagiarism is when you copy a whole assignment verbatim from a third party and present it as your own. That includes handing in essays that were written by someone else or submitting papers that you found online.

2. Direct Plagiarism:

Direct plagiarism is when you copy entire sentences or paragraphs from another person's writing and use them in your own work.
The assignment typically includes some of your original writing as well, as opposed to pure plagiarism. But taking content verbatim from outside sources and not crediting them is against academic plagiarism rules.

3. Paraphrasing Without Citing Sources:

Plagiarism regulations at colleges prohibit rewriting someone else's words without giving credit to the original author. This involves modifying a few words in someone else's sentences and stealing ideas without giving credit.

4. False Citations:

Falsifying citations may appear insignificant in comparison to plagiarizing entire paragraphs, yet it still counts as plagiarism. Making up quotes or data points constitutes a fake citation and is prohibited by plagiarism laws.
One of the most prevalent instances of unintentional plagiarism is the use of false citations. Always keep track of your sources so you can properly credit each one in your writing.

5. Self-Plagiarism:

plagiarize yourself means you can't turn in the same paper in two classes. It also implies that you cannot incorporate content from previous tasks into your current work.

2. Text Similarity

2.2 Semantic Similarity

Semantic Similarity, or Semantic Textual Similarity, is a task in the area of Natural Language Processing (NLP) that scores the relationship between texts or documents using a defined metric. Semantic Similarity has various applications, such as information retrieval, text summarization, sentiment analysis, etc.

In contrast to lexicographical similarity, semantic similarity is a measure of distance between things based on how similar their meanings or semantic contents are. It is defined over a set of texts or phrases.

These are mathematical instruments that are used to gauge the strength of the semantic connection between language units, concepts, or instances by a numerical description derived from a comparison of the data evidencing their meaning or outlining their characteristics.

Semantic relatedness and semantic similarity are frequently used interchangeably. While semantic similarity solely takes into account "is a" relations, semantic relatedness encompasses all relationships between two concepts. For instance, "car" is related to "road" and "driving" and is similar to "bus."

3. PROJECT IDEA

in order to apply the concepts of NLP that have been mentioned above we developed an online platform that is a text similarity checker for checking the similarity between the graduation project from their abstract to calculate the percentage of similarity between the rest of the project.

The system mainly divided into 3 part of function :

UPLOAD FILES | PLAGIARISM | SEMANTIC

UPLOADING FILE: After login, the users can upload these files in the system from the upload button, after the user selects the file, it will be saved in the system.

PLAGIARISM: in this part of the system, the user can check the similarity between every 2 documents that he has uploaded to the system from and check the present of same words and sentences in both files.

SEMANTIC SIMILARITY: The aim of determining the semantic similarity measures between two sets of sentences that have been entered by the user and finding the degree of relevance by matching the words meaning, which are conceptually similar but not necessarily lexicographically similar .

4. Methodology

4.1 Plagiarism similarity

- **Text Pre-Processing**

In day to day practice, information is being gathered from multiple sources be it web, document or transcription from audio, this information may contain various types of garbage values, noisy text, encoding. This needs to be cleaned in order to perform further tasks of NLP. In this preprocessing phase, it should include removing non-ASCII values, special characters, HTML tags, stop words, raw format conversion and so on

- **Feature Extraction**

To convert the text data into a numeric format, text data needs to be encoded. Various encoding techniques are widely being used to extract the word-embeddings from the text data such techniques are bag-of-words, TF-IDF, word2vec

- **Vector Similarity**

Once we will have vectors of the given text chunk, to compute the similarity between generated vectors, statistical methods for the vector similarity can be used. Such techniques are cosine similarity, Euclidean distance, Jaccard distance, word mover's distance. Cosine similarity is the technique that is being widely used for text similarity

- **Decision Function**

From the similarity score, a custom function needs to be defined to decide whether the score classifies the pair of chunks as similar or not. Cosine similarity returns the score between 0 and 1 which refers 1 as the exact similar and 0 as the nothing similar from the pair of chunks. In regular practice, if the similarity score is more than 0.5 than it is likely to similar at a somewhat level. But, this can vary based on the application and use-case

4. Methodology

4.2 Plagiarism similarity Libraries

- **Cousin similarity**

Cosine similarity is a metric used to measure how similar the documents are irrespective of their size. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space

The cosine similarity is advantageous because even if the two similar documents are far apart by the Euclidean distance (due to the size of the document), chances are they may still be oriented closer together. The smaller the angle, higher the cosine similarity.

- **TFIDF Vectorizer**

TFIDF works by proportionally increasing the number of times a word appears in the document but is counterbalanced by the number of documents in which it is present. Hence, words like 'this', 'are' etc., that are commonly present in all the documents are not given a very high rank

4. Methodology

4.3 Semantic Similarity

- Sentences transformer :

The most straightforward and effective method now is to use a powerful model (e.g. transformer) to encode sentences to get their embeddings and then use a similarity metric (e.g. cosine similarity) to compute their similarity score. The similarity score indicates whether two texts have similar or more different meanings. This post will show how to implement Semantic Similarity using Transformers, which is a powerful NLP architecture that has resulted in state-of-the-art performance for various NLP tasks.

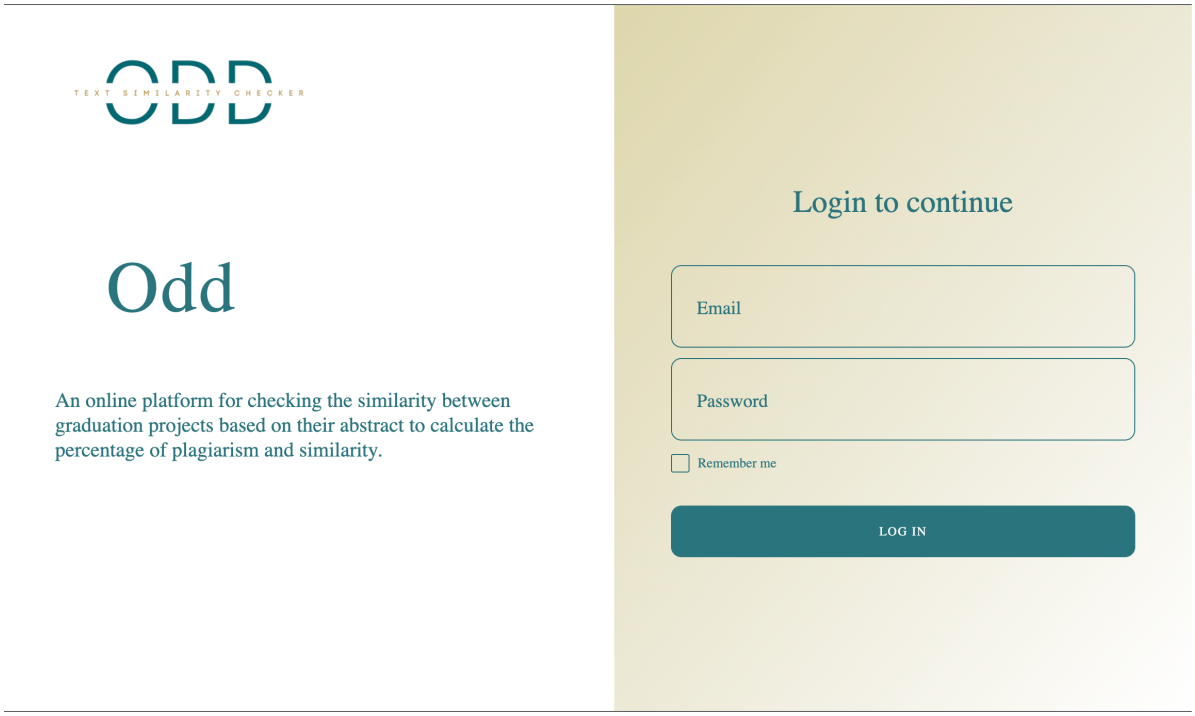
- BERT model :

Sentence Similarity is the task of determining how similar two texts are. Sentence similarity models convert input texts into vectors (embeddings) that capture semantic information and calculate how close (similar) they are between them. This task is particularly useful for information retrieval and clustering/grouping.

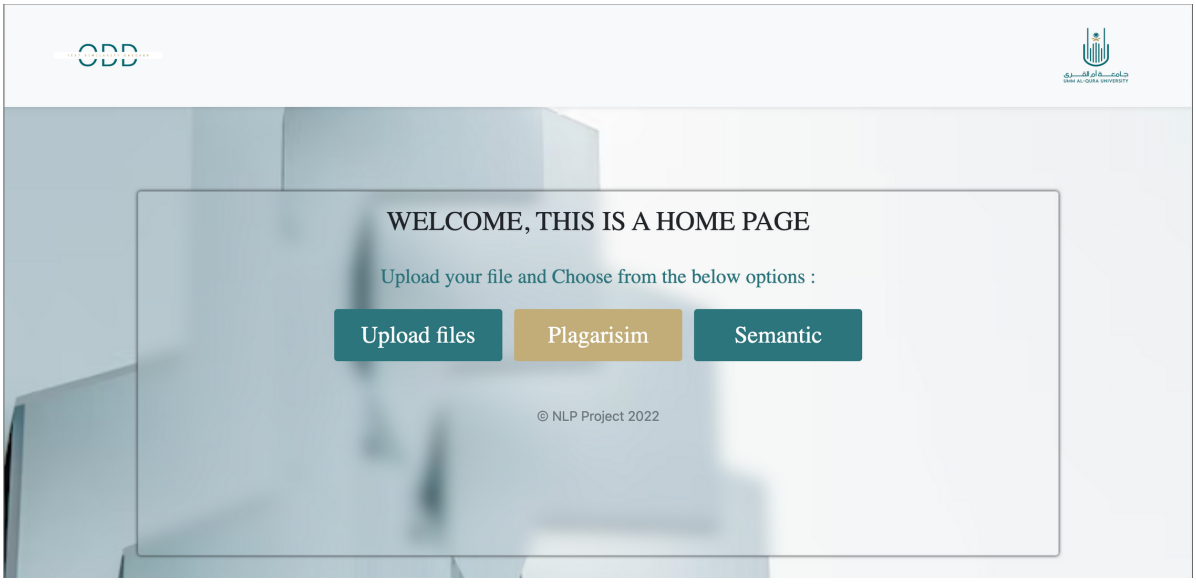
in our project, we use the BERT model (bert-base-nli-mean-tokens) to check the semantic similarity since BERT is the MVP of NLP. And a big part of this is down to BERTs ability to embed the meaning of words into densely packed vectors.

is called dense vectors because every value within the vector has a value and has a reason for being that value — this is in contrast to sparse vectors, such as one hot encoded vector where the majority of values are 0.
BERT is great at creating these dense vectors, and each encoder layer (there are several) outputs a set of dense vectors.

5. INTERFACES

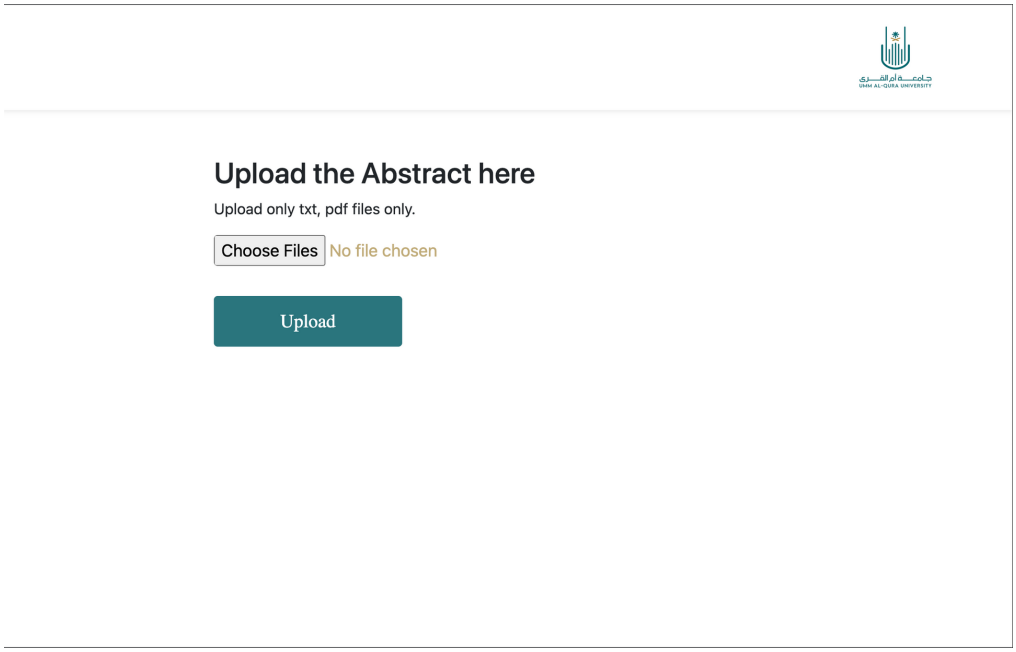


ODD Log in interface

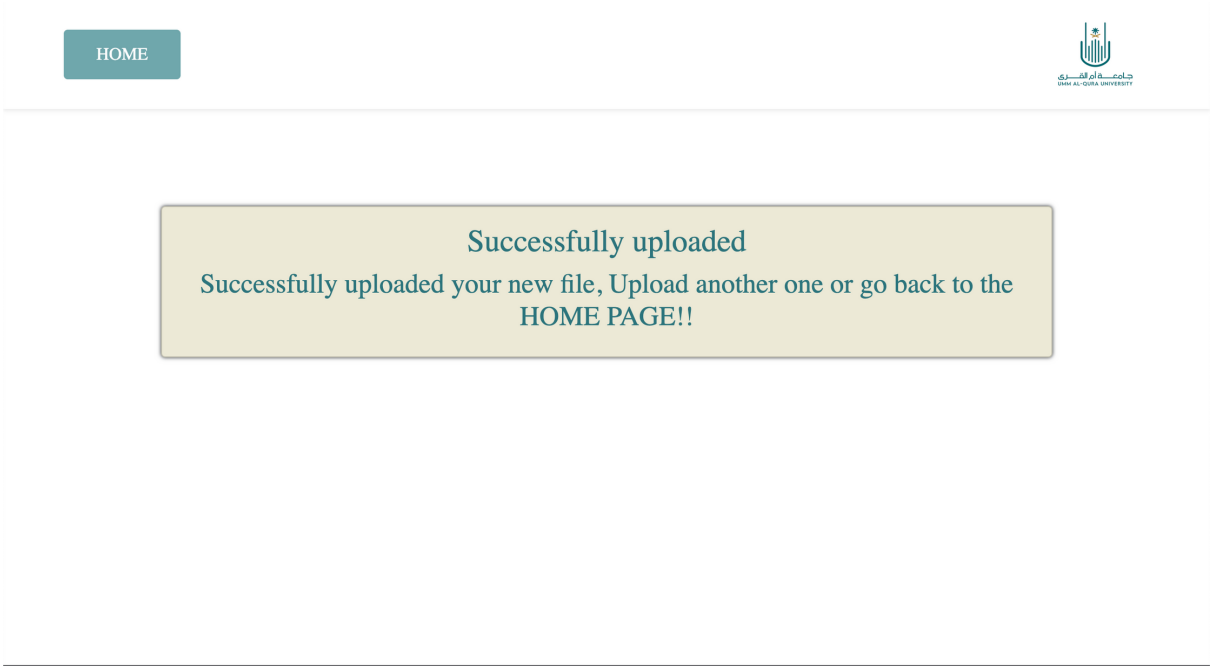


ODD Home interface

5. INTERFACES

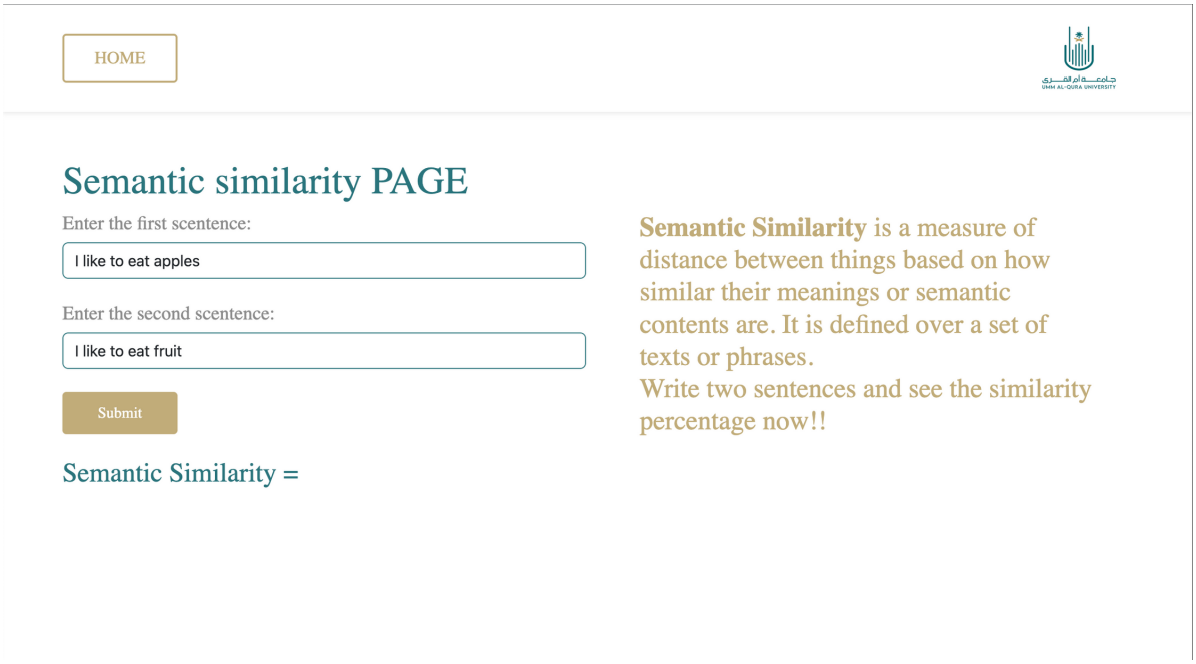


ODD Uploading files interface



ODD Uploading files's respond interface

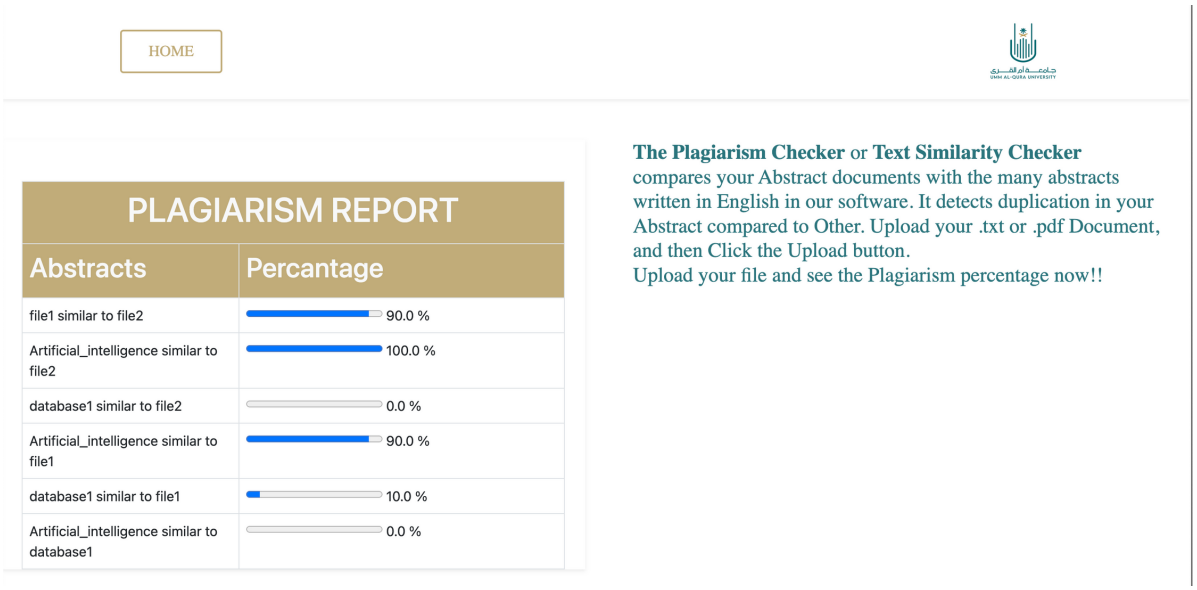
5. INTERFACES



ODD Semantic similarity interface

ODD Semantic similarity's result interface

5. INTERFACES



ODD Plagiarism Report interface

6. IMPLEMENTATION

Main app.py file that use flask and python to run the web interfaces

semantic.py* similar.py utils.py app.py

```python
import os
from flask import Flask, render_template, redirect, flash, request, send_from_directory
from werkzeug.utils import secure_filename
from utils import *
import glob
from similar import check_plagiarism
from semantic import seman

app = Flask(__name__)
app.config['UPLOADS_FOLDER'] = UPLOADS_FOLDER
app.config['SECRET_KEY'] = 'my secret'

@app.route('/', methods=["GET", "POST"])
def login():
    return render_template('Log-In.html')

@app.route('/home', methods=["GET", "POST"])
def home():
    return render_template('home.html')

@app.route('/index', methods=["GET", "POST"])
def index():
    if request.method == "GET":
        return render_template('index.html')

    if not 'file' in request.files:
        flash('No file part in request')
        return redirect(request.url)

    files = request.files.getlist('file')

    for file in files:
        if file.filename == '':
            flash('No file uploaded')
            return redirect(request.url)

        if file_valid(file.filename):
            filename = secure_filename(file.filename)
            file.save(os.path.join(app.config['UPLOADS_FOLDER'], filename))
            return render_template('second.html')
        else:
            flash('Invalid file type')
            return redirect(request.url)



@app.route('/check',methods=['POST', 'GET'])
def checker():
    similarity=check_plagiarism()
    return render_template('checking.html',similarity=similarity)


@app.route('/semantic', methods=['POST', 'GET'])
def sem():
    return render_template('semantic.html')

@app.route('/result',methods=['POST', 'GET'])
def result():
    output = request.form.to_dict()
    print(output)
    first = output["first"]
    second=output["second"]
    score=seman(first=first,second=second)

    return render_template('semantic.html', score=score)
```

6. IMPLEMENTATION

Semantic.py that use BERT model for the semantic similarity between 2 text

```python
from sentence_transformers import SentenceTransformer
from sklearn.metrics.pairwise import cosine_similarity

def seman(first,second):
    sentences = [
        first,
        second

    ]
    model = SentenceTransformer('bert-base-nli-mean-tokens')

    sentence_embeddings = model.encode(sentences)
    score= cosine_similarity(
        [sentence_embeddings[0]],
        sentence_embeddings[1:]
    )
    return (score[0][0]*100)
```

similar.py that calculate the plagiarism percentage for each pair of files

```python
import os
import urllib.parse
import json
import sys

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

student_files = [doc for doc in os.listdir() if doc.endswith('.txt')]
student_notes = [open(_file, encoding='utf-8').read()
                 for _file in student_files]

def vectorize(Text): return TfidfVectorizer().fit_transform(Text).toarray()
def similarity(doc1, doc2): return cosine_similarity([doc1, doc2])

vectors = vectorize(student_notes)
s_vectors = list(zip(student_files, vectors))

def check_plagiarism():
    plagiarism_results = {}
    global s_vectors
    for student_a, text_vector_a in s_vectors:
        new_vectors = s_vectors.copy()
        current_index = new_vectors.index((student_a, text_vector_a))
        del new_vectors[current_index]
        for student_b, text_vector_b in new_vectors:
            sim_score = similarity(text_vector_a, text_vector_b)[0][1]
            if(sim_score > 0):
                sim_score = round(sim_score, 1)
                student_pair = sorted((os.path.splitext(student_a)[0], os.path.splitext(student_b)[0]))
                res = (student_pair[0]+' similar to '+ student_pair[1])
                plagiarism_results[res] = sim_score*100

    return plagiarism_results

if __name__ == "__main__":
    print(check_plagiarism())
```

utils.py that specify the accepted form of files and the path to store the uploaded files

```python
ALLOWED_EXTENSIONS = ['txt','pdf']
#UPLOADS_FOLDER = 'uploads/abstract'
UPLOADS_FOLDER = '/Users/saharali/Desktop/Flask-Series-files-in-flask copy'


def file_valid(file):
    return '.' in file and \
        file.rsplit('.', 1)[1] in ALLOWED_EXTENSIONS
```

6. IMPLEMENTATION

checking.html is the page that display the plagiarism report

checking.html | home.html | index.html | Log-In.html | second.html | semantic.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 href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/css/bootstrap.min.css" rel="stylesheet"
    integrity="sha384-+0n0xVW2eSR5OomGNYDnhzAbDsOXxcvSN1TPprVMTNDbiYZCxYbOOl7+AMvyTG2x" crossorigin="anonymous" />
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.3.0/font/bootstrap-icons.css" />
  <link href="https://api.mapbox.com/mapbox-gl-js/v2.1.1/mapbox-gl.css" rel="stylesheet" />
  <link rel="stylesheet" href="style.css" />
  <title>Plagiarism</title>
</head>

<body>
  <form action="/home" method="post" enctype="multipart/form-data">

    <input type="submit" value="HOME">
  </form>

  <!-- Showcase -->
  <section class="text-light p-5 p-lg-0 pt-lg-5 text-center text-sm-start">
    <div class="container">
      <div class="align-items-center justify-content-between">
        <div>
          <table class="table table-bordered">
            <thead style="position: sticky;top: 0" class="bg-danger text-light">
              <tr>
                <th colspan="2" class="text-center">
                  <h1>PLAGIARISM REPORT</h1>

                </th>
              </tr>
              <tr>
                <th scope="col" width="40%">
                  <h2>Abstracts</h2>
                </th>
                <th scope="col" width="60%">
                  <h2>Percantage</h2>
                </th>
                 </th>
              </tr>
              <thead>
              <tbody>
                {% for key, value in similarity.items() %}
                <tr>
                  <td> {{ key }} </td>
                  <td><progress id="file" value={{ value }} max="100"></progress> {{ value }} % </td>
                </tr>
                {% endfor %}
              </tbody>

            </table>
          </div>
        </div>
      </div>
    </section>

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/js/bootstrap.bundle.min.js"
      integrity="sha384-gtEjrD/SeCtmISkJkNUaaKMoLD0//ElJ19smozuHV6z3Iehds+3Ulb9Bn9Plx0x4" crossorigin="anonymous">
      </script>
</body>

</html>
```

6. IMPLEMENTATION

home.html is the page that display the three functions options to the user

checking.html | home.html | index.html | Log-In.html | second.html | semantic.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>HOME PAGE</title>
</head>
<body>
    <h>THIS IS A HOME PAGE , WELCOME</h>
    <form>
        <input type="submit" value="Upload files" formaction="/index">
        <input type="submit" value="Plagarisim" formaction="/check">
        <input type="submit" value="Semantic" formaction="/semantic">
    </form>
</body>
</html>
```

index.html is the uploading files html page that require a pdf or txt file to be uploaded

checking.html | home.html | index.html | Log-In.html | second.html | semantic.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>Files in Flask</title>
  <style>
    .container {
      width: 80%;
      margin: 2rem auto;
    }

    input {
      display: block;
      margin-bottom: 2rem;
      font-size: 18px;
    }

    input[type="submit"] {
      background-color: rgb(17, 101, 78);
      color: #fff;
      padding: .8rem;
      border-radius: 10px;
    }

    .error {
      color: #ff0000;
    }
  </style>
</head>
<body>
  <div class="container">
    <h1>Upload the Abstract here</h1>
    <p>Upload only txt, pdf files only.</p>
    <form action="/index" method="post" enctype="multipart/form-data">
      <input type="file" name="file" required multiple>
      <input type="submit" value="Upload">
    </form>

    {% with messages = get_flashed_messages() %}
      {% for msg in messages %}
        <p class="error">{{ msg }}</p>
      {% endfor %}
    {% endwith %}
  </div>
</body>
</html>
```

6. IMPLEMENTATION

Log-in html code

checking.html | home.html | index.html | Log-In.html | second.html | semantic.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Log In</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="icon" type="user.png" href="/Users/saharali/Desktop/Flask-Series-files-in-flask copy/user.png">
<!--===============================================================================================-->
<!--===============================================================================================-->
    <link rel="stylesheet" type="text/css" href="/Users/saharali/Desktop/Flask-Series-files-in-flask copy/vendor/bootstrap/css/bootstrap.min.css">
<!--===============================================================================================-->
    <link rel="stylesheet" type="text/css" href="fonts/font-awesome-4.7.0/css/font-awesome.min.css">
<!--===============================================================================================-->
    <link rel="stylesheet" type="text/css" href="fonts/Linearicons-Free-v1.0.0/icon-font.min.css">
<!--===============================================================================================-->
    <link rel="stylesheet" type="text/css" href="vendor/animate/animate.css">
<!--===============================================================================================-->
    <link rel="stylesheet" type="text/css" href="vendor/css-hamburgers/hamburgers.min.css">
<!--===============================================================================================-->
    <link rel="stylesheet" type="text/css" href="vendor/animsition/css/animsition.min.css">
<!--===============================================================================================-->
    <link rel="stylesheet" type="text/css" href="vendor/select2/select2.min.css">
<!--===============================================================================================-->
    <link rel="stylesheet" type="text/css" href="vendor/daterangepicker/daterangepicker.css">
<!--===============================================================================================-->
    <link rel="stylesheet" type="text/css" href="/Users/saharali/Desktop/Flask-Series-files-in-flask copy/css/util.css">
    <link rel="stylesheet" type="text/css" href="/Users/saharali/Desktop/Flask-Series-files-in-flask copy/css/Log-In.css">
<!--===============================================================================================-->

</head>
<body>

    <div class="limiter">
        <div class="container-login100">
            <div class="wrap-login100">
                <form action="/home" method="post" enctype="multipart/form-data" class="login100-form validate-form">
                    <span class="login100-form-title p-b-43">
                        Login to continue
                    </span>


                    <div class="wrap-input100 validate-input" data-validate = "Valid email is required: ex@abc.xyz">
                        <input class="input100" type="text" name="email">
                        <span class="focus-input100"></span>
                        <span class="label-input100">Email</span>
                    </div>


                    <div class="wrap-input100 validate-input" data-validate="Password is required">
                        <input class="input100" type="password" name="pass">
                        <span class="focus-input100"></span>
                        <span class="label-input100">Password</span>
                    </div>

                    <div class="flex-sb-m w-full p-t-3 p-b-32">
                        <div class="contact100-form-checkbox">
                            <input class="input-checkbox100" id="ckb1" type="checkbox" name="remember-me">
                            <label class="label-checkbox100" for="ckb1">
                                Remember me
                            </label>
                        </div>


                    </div>

                    <input type="submit" value="LOG IN">



                </form>

                <div class="login100-more" style="background-color: #f8f2e4;">
                            <div class="backgroundTexts">
                        <span class="backgroundTitle">XXXXXXXX </span><br>
                        <span class="backgroundP"><br>XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX<br>
                                                  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX <br>
                                                  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX</span>
                </div>
                </div>

            </div>
        </div>
    </div>




<!--===============================================================================================-->
    <script src="vendor/jquery/jquery-3.2.1.min.js"></script>
<!--===============================================================================================-->
    <script src="vendor/animsition/js/animsition.min.js"></script>
<!--===============================================================================================-->
    <script src="vendor/bootstrap/js/popper.js"></script>
    <script src="vendor/bootstrap/js/bootstrap.min.js"></script>
<!--===============================================================================================-->
    <script src="vendor/select2/select2.min.js"></script>
<!--===============================================================================================-->
    <script src="vendor/daterangepicker/moment.min.js"></script>
    <script src="vendor/daterangepicker/daterangepicker.js"></script>
<!--===============================================================================================-->
    <script src="vendor/countdowntime/countdowntime.js"></script>
<!--===============================================================================================-->
    <script src="LogInJs.js"></script>

</body>
</html>
```

6. IMPLEMENTATION

second.html display the respond from the uploading file function

checking.html | home.html | index.html | Log-In.html | second.html | semantic.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>Document</title>
</head>
<body>
    <h1>Successfully uploaded</h1>
    <p>Successfully uploaded your new file, Upload another one or go back to the HOME PAGE!!</p>

    <form action="/home" method="post" enctype="multipart/form-data">

        <input type="submit" value="home">
      </form>


</body>
</html>
```

semantic.html is the page where the user enters 2 text then get the semantic similarity percentage

checking.html | home.html | index.html | Log-In.html | second.html | semantic.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>Document</title>
</head>
<body>
    <h1> Semantic similarity PAGE</h1>

    <form action="/home" method="post" enctype="multipart/form-data">

        <input type="submit" value="HOME">
      </form>


    <form class="grid" action="/result" method="POST">



        <label for="first">Enter the first scentence:</label>
        <input type="text" id="first" name="first">
<br>
        <label for="second">Enter the second scentence:</label>
        <input type="text" id="second" name="second">

       <input class="file_submit" type="Submit">



        <h3 style="color: blue;" class="let_space">Semantic Similarity =  </h3>
        <p>{{score}}</p>



      </form>

</body>
</html>
```