Anuj Siddharth: 23011101024
Mohammad Farhan: 23011101079

Script Plagiarism Checker – Project Documentation

1. Analysis of Challenges

Working with movie or play scripts poses unique challenges for text-based analysis:

Unstructured Format: Scripts often mix dialogue, scene directions, and stage cues inconsistently. This irregularity makes it hard to isolate meaningful content for NLP-based comparisons.

Noise in Text: Elements like character names, timestamps, and non-speech notations introduce semantic noise, skewing similarity scores.

Writing Style Variation: Two scripts may convey identical content using vastly different vocabulary and syntax, which makes surface-level similarity measures less effective.

Stopword Distortion: High-frequency words such as *"and", "the", "is"* have little semantic value but appear frequently, reducing the precision of matching.

Scalability: With **N** uploaded scripts, the system performs N(N−1)2\frac{N(N - 1)}{2} pairwise comparisons. This quadratic growth impacts performance with large inputs.

2. Application Selection

This project was implemented using **Streamlit** as a web-based front end. It allows:

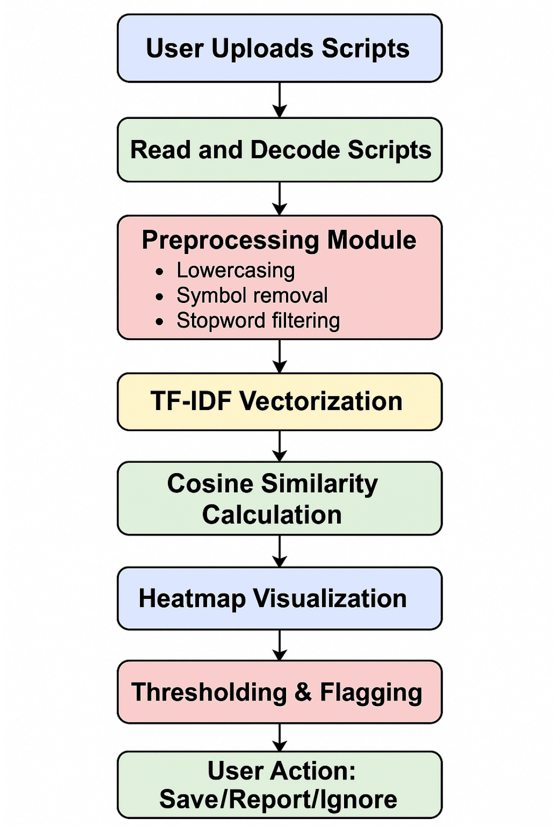
- Plagiarism detection via pairwise similarity of uploaded text files.
- Multiple script uploads and batch processing.
- Visualization of similarity through heatmaps.

- Suggestions and warnings when suspicious similarities are detected.

The application is lightweight and user-friendly, aimed at teachers, reviewers, and academic institutions.

---

3. Architecture Diagram



System Flow:

1. **File Upload** → Multiple `.txt` files selected by user.
2. **File Decoding** → All scripts are read in `UTF-8`.
3. **Preprocessing** → Lowercasing, regex cleaning, stopword removal.
4. **Vectorization** → TF-IDF transformation of clean text.

5. **Similarity Matrix** → Pairwise cosine similarity of TF-IDF vectors.
6. **Heatmap Plot** → Seaborn-based matrix visualization.
7. **Flagging & Thresholding** → If similarity > 0.55, it's flagged.
8. **User Notification** → Guidance provided if plagiarism suspected.

---

4. Module Description

4.1 Preprocessing

```
def preprocess_script(script_text):
    script_text = script_text.lower()
    script_text = re.sub(r"[^a-z\s]", "", script_text)
    tokens = script_text.split()
    stop_words = text.ENGLISH_STOP_WORDS
    tokens = [word for word in tokens if word not in stop_words]
    return " ".join(tokens)
```

Explanation:

Let TT be the raw text string.

- **Lowercasing** maps T→T′T \to T', where all characters c∈T′c \in T' are c=lower(c)c = \text{lower}(c).
- **Regex Filter** removes characters c∉[a−z]∪{space}c \notin [a-z] \cup \{\text{space}\}.
- **Stopword Removal** uses set S⊂English stopwordsS \subset \text{English stopwords} and removes all t∈T′t \in T' where t∈St \in S.

Why it works: This simplifies the input space XX, reducing noise and dimensionality before vectorization.

Advanced options:

- Lemmatization: "running"→"run"\text{"running"} \to \text{"run"}

- Named Entity Recognition (NER) to remove proper nouns (e.g., character names).
- Subword tokenization with BERT.

4.2 Vectorization

```
def vectorize_scripts(scripts):
    vectorizer = TfidfVectorizer()
    return vectorizer.fit_transform(scripts)
```

TF-IDF Formula: Let:

- ft,df_{t,d}: frequency of term tt in document dd
- NN: total number of documents
- dftdf_t: number of documents containing term tt

Then the **TF-IDF weight** of term tt in document dd is:

TF-IDF(t,d)=ft,d⋅log(N1+dft)\text{TF-IDF}(t, d) = f_{t,d} \cdot \log\left(\frac{N}{1 + df_t}\right)

Why it helps: It emphasizes rare but important terms and suppresses ubiquitous terms.

Alternatives:

- **CountVectorizer**: Based only on frequency, no down-weighting.
- **Word2Vec/Doc2Vec**: Captures word meaning and order.
- **Transformer Embeddings (e.g., BERT)**: Context-aware vectorization.

4.3 Similarity Calculation

```
def calculate_similarity(tfidf_matrix):
    return cosine_similarity(tfidf_matrix)
```

Cosine Similarity:

Given two document vectors $\vec{u}, \vec{v}$, the similarity is:

$$\cos(\theta) = \frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\| \|\vec{v}\|}$$

Where:

- $\cdot$: dot product
- $\|\vec{u}\|$: Euclidean norm of $\vec{u}$

Why cosine?

- Range: $[0, 1]$ for non-negative vectors.
- Invariant to vector length → good for scripts of varying lengths.
- Efficient on sparse matrices.

4.4 Visualization

```
def display_heatmap(similarity_matrix, filenames):
    fig, ax = plt.subplots(figsize=(10, 8))
    sns.heatmap(similarity_matrix, xticklabels=filenames, yticklabels=filenames,
                cmap="YlGnBu", annot=True, fmt=".2f", ax=ax)
    st.pyplot(fig)
```

- Matrix $S \in \mathbb{R}^{N \times N}$ is visualized using a color gradient.
- Diagonal elements $S_{ii} = 1.0$ (self-similarity).
- Annotated values aid in interpreting exact percentage matches.

Why use heatmaps:

- High interpretability for educators.
- Clearly shows outliers (potential plagiarism).

4.5 Plagiarism Detection

```
threshold = 0.55
for i in range(n):
    for j in range(i + 1, n):
        if similarity_matrix[i][j] > threshold:
            flagged.append((filenames[i], filenames[j], similarity_matrix[i][j]))
```

This checks:

Sij>τwhere τ=0.55S_{ij} > \tau \quad \text{where } \tau = 0.55

Only upper triangle (i<j)(i < j) is considered due to symmetry.

Why 0.55? Empirically tuned to reduce false positives and maintain sensitivity.

Extensions:

- Dynamic threshold slider.
- Rank suspicious cases by descending similarity.

---

5. Data Selection and Preprocessing

Input Format:

- Text files (`.txt`) in UTF-8 encoding.
- Typically: film, play, or scene scripts.

Preprocessing Summary:

- Lowercase transformation
- Regex-based symbol cleaning
- Stopword filtering

Advanced Enhancements:

- Part-of-Speech (POS) filtering (retain only nouns, verbs).
- Coreference resolution.
- Named Entity Removal (strip names/locations).

---

6. Performance Evaluation

Primary Metric: Cosine Similarity ∈[0,1]\in [0, 1]

- **0**: No overlap
- **1**: Exact match

Chosen threshold τ=0.55\tau = 0.55 balances:

- **Precision** (true positives / detected positives)
- **Recall** (true positives / actual positives)

Suggestions for Improvement:

- Add ground-truth plagiarized scripts to compute precision, recall, F1-score.
- Compare:
    - TF-IDF
    - BERT
    - Universal Sentence Encoder (USE)

---

7. Conclusion and Future Work

Conclusion

- The app effectively flags potential plagiarism using classical NLP.
- Simple interface ensures usability by non-technical users.
- TF-IDF and cosine similarity provide a good baseline.
- Visualization aids quick decision-making.

Future Work

- Use **SBERT** or **BERT** for contextual embedding.
- Add user-controlled similarity threshold slider.
- Auto-generate **PDF reports** for flagged cases.
- Highlight **similar passages** using sequence matching.

8.Code:

```
import streamlit as st
import numpy as np
import re
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.feature_extraction import text  # for stopwords

# ---------- Preprocessing ----------
def preprocess_script(script_text):
    script_text = script_text.lower()
    script_text = re.sub(r"[^a-z\s]", "", script_text)
    tokens = script_text.split()
    stop_words = text.ENGLISH_STOP_WORDS
    tokens = [word for word in tokens if word not in stop_words]
    return " ".join(tokens)

# ---------- Vectorization ----------
```

```
def vectorize_scripts(scripts):
    vectorizer = TfidfVectorizer()
    return vectorizer.fit_transform(scripts)

# ---------- Similarity Calculation ----------
def calculate_similarity(tfidf_matrix):
    return cosine_similarity(tfidf_matrix)

# ---------- Visualization ----------
def display_heatmap(similarity_matrix, filenames):
    fig, ax = plt.subplots(figsize=(10, 8))
    sns.heatmap(similarity_matrix, xticklabels=filenames, yticklabels=filenames,
                cmap="YlGnBu", annot=True, fmt=".2f", ax=ax)
    st.pyplot(fig)

# ---------- Main App ----------
st.title("🎬 Script Plagiarism Detector")
st.write("Upload multiple movie scripts in `.txt` format to compare their similarity.")

uploaded_files = st.file_uploader("Choose script files", type="txt", accept_multiple_files=True)

if uploaded_files:
    if len(uploaded_files) < 2:
        st.warning("Please upload **at least two** script files to check for plagiarism.")
    else:
        raw_scripts = []
        filenames = []

        for file in uploaded_files:
            text_data = file.read().decode("utf-8")
            raw_scripts.append(preprocess_script(text_data))
            filenames.append(file.name)

        st.success("Scripts uploaded and preprocessed successfully.")

        with st.spinner("Calculating similarity..."):
            tfidf_matrix = vectorize_scripts(raw_scripts)
            similarity_matrix = calculate_similarity(tfidf_matrix)

        st.subheader("🔍 Similarity Heatmap")
        display_heatmap(similarity_matrix, filenames)

        st.subheader("⚠️ Potential Plagiarism Cases")
        threshold = 0.55
        flagged = []
        n = len(filenames)
        for i in range(n):
            for j in range(i + 1, n):
                if similarity_matrix[i][j] > threshold:
                    flagged.append((filenames[i], filenames[j], similarity_matrix[i][j]))

        if flagged:
            for file1, file2, score in flagged:
                percentage = score * 100
                st.write(f"📝 **'{file1}' and '{file2}' are {percentage:.0f}% similar.** This may indicate
potential plagiarism.")

            st.markdown("---")
            st.subheader("📢 What to do if plagiarism is detected?")
            st.markdown("""
If you suspect plagiarism:
- 📁 **Save the similarity report** or screenshot the results.
- 🧑‍🏫 **Inform your course instructor** or TA with the report and files involved.
- 🛑 **Avoid making accusations without evidence** — let the academic team investigate.
- 📧 Many institutions have a plagiarism reporting system or email — check your university's
academic integrity policies.
""")
        else:
            st.info("✅ No suspicious similarity scores found above threshold.")
```

9.Technologies Used

- **Python 3**
- **Streamlit** (for the user interface)
- **Scikit-learn** (for TF-IDF and cosine similarity)
- **Matplotlib & Seaborn** (for similarity heatmap visualization)
- **re** and **nltk** (for text preprocessing)

10.How It Works

1. **File Upload**
   Users upload exactly two `.txt` files (scripts). The app will process and compare them.

2. **Text Preprocessing**
   Each script is cleaned by:
   - Lowercasing
   - Removing punctuation and numbers
   - Removing stopwords
   - Tokenizing

3. **Vectorization**
   The cleaned scripts are converted into **TF-IDF vectors** using `TfidfVectorizer`.

4. **Similarity Calculation**
   Cosine similarity is calculated between the two vectorized scripts to measure how alike they are.

5. **Visualization & Result**
   - A heatmap shows the pairwise similarity.
   - If similarity exceeds a set **threshold of 0.55**, the app flags potential plagiarism.

6. **Actionable Message**
   If plagiarism is suspected,the app provides a message guiding users on how to report it.

11.Sample Scripts

`script1.txt`

```
INT. LIVING ROOM - NIGHT
```

A clock ticks loudly in the silence. JAMES, late 20s, sits on the couch staring at a blank TV screen. His phone vibrates. He doesn’t move.

Suddenly, a KNOCK at the door. He turns his head slowly.

JAMES
(quietly)
Who could that be?

He walks to the door and opens it to find LISA, early 30s, holding a small box.

LISA
You forgot this.

She hands him the box. He stares at it, confused.

JAMES
I thought I lost it...

LISA
You did. But I found it — in the drawer where you hide things.

She smiles faintly and walks away into the night.

JAMES looks at the box, then closes the door gently.

---

script2.txt

INT. LIVING ROOM — NIGHT

A clock ticks in the quiet room. JAMES, in his late 20s, is slouched on the couch, staring at a blank TV. His phone buzzes. He ignores it.

There's a soft KNOCK at the door. He looks toward it.

JAMES
(low voice)
Who’s that?

He slowly opens the door. LISA, early 30s, stands there with a small box in hand.

LISA
You left this behind.

He takes the box, puzzled.

JAMES
I thought it was gone...

LISA
You hid it, remember? I found it where you usually stash things.

She walks off into the dark. JAMES holds the box, lost in thought, and closes the door behind him.

---

script3.txt

EXT. FOREST TRAIL – MORNING

The sun peeks through the tall pine trees. Birds chirp. EMMA, 40s, jogs with her dog, MAX.

EMMA
Keep up, Max!

The dog barks and runs ahead. Suddenly, Max stops and growls at something in the bushes.

EMMA
What is it, boy?

```
She walks over, pushes the leaves aside — revealing a rusted metal box
buried in the dirt.

She kneels, opens it slowly. Inside is a faded photograph and a key.

EMMA
What on earth...?
```

---

script4.txt

```
INT. COFFEE SHOP — DAY

The café is buzzing with quiet chatter. ALEX, early 20s, nervously
taps a pen at a table by the window.

MIRA, late 20s, walks in and spots him. She smiles and approaches.

MIRA
Sorry I’m late. Traffic was chaos.

ALEX
It’s okay. I thought maybe you changed your mind.

MIRA
Not a chance. You said you had something important to say?

ALEX takes a deep breath and pulls out a folded letter.

ALEX
It’s... about your brother. He didn’t die in that crash.

MIRA stares at him, speechless.
```

12.What Happens When Plagiarism is Detected?

When the app finds two scripts with a similarity score above **0.55**, it will:

- Clearly display a warning.
- Show how much the scripts are alike (e.g., *"The scripts 'script2.txt' and 'script1.txt' are 61% similar"*).
- Display a message advising the user to take action.

Suggested Action Message:

> ⚠️ **If you believe plagiarism has occurred, please report it to your course instructor or academic integrity board.**
> You may also attach the scripts and this similarity report in your complaint.

13.Requirements

Make sure these libraries are installed:

```
pip install streamlit scikit-learn matplotlib seaborn nltk
```

Also, initialize NLTK stopwords once:

python
CopyEdit
```
import nltk
nltk.download('stopwords')
```

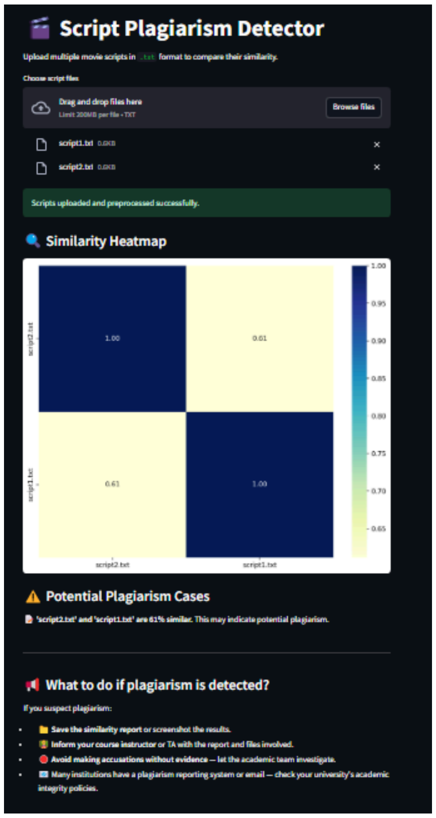
14.Launching the App

Navigate to the project directory and run:

bash
CopyEdit
```
streamlit run script_plagiarism_streamlit.py
```

15.Sample Output:

Case 1:



🎬 Script Plagiarism Detector

Upload multiple movie scripts in `.txt` format to compare their similarity.

Choose script files

Drag and drop files here
Limit 200MB per file • TXT

script1.txt 0.6KB

script2.txt 0.6KB

Scripts uploaded and preprocessed successfully.

🔍 Similarity Heatmap

⚠ Potential Plagiarism Cases

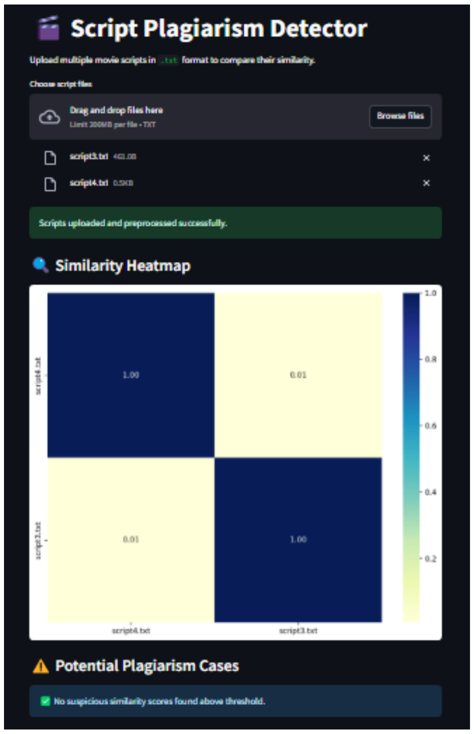
📝 'script2.txt' and 'script1.txt' are 61% similar. This may indicate potential plagiarism.

📢 What to do if plagiarism is detected?

If you suspect plagiarism:

- 📁 Save the similarity report or screenshot the results.
- 🧑‍🏫 Inform your course instructor or TA with the report and files involved.
- 🔴 Avoid making accusations without evidence — let the academic team investigate.
- 📧 Many institutions have a plagiarism reporting system or email — check your university's academic integrity policies.

Case 2:



🎬 Script Plagiarism Detector

Upload multiple movie scripts in `.txt` format to compare their similarity.

Choose script files

Drag and drop files here
Limit 200MB per file • TXT

script3.txt 461.0B

script4.txt 0.5KB

Scripts uploaded and preprocessed successfully.

🔍 Similarity Heatmap

⚠ Potential Plagiarism Cases

✅ No suspicious similarity scores found above threshold.

16. Key Learnings

Text Preprocessing

- Crucial to normalize input to avoid inflated similarity from common words.

TF-IDF + Cosine

- Helps represent unstructured scripts as numerical vectors.
- Cosine metric allows intuitive similarity comparison.

Thresholding Logic

- Importance of tuning similarity thresholds to reduce false alarms.

Streamlit Front-End

- Enabled rapid development of a functional UI.

Ethics & Fairness

- Tools like this reinforce academic honesty and discourage unethical submissions.

Finally, we would like to express our sincere gratitude to our university and faculty for providing us with this opportunity to explore and implement our ideas through this project.