DOCUMENT PLAGIARISM DETECTION

A Report on Course end project Submitted in the Partial Fulfillment of the Requirements for the Award of the Degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING (AI&ML)

Submitted By

POOJALA JAIVARDHAN	24881A66G7

Under the Esteemed Guidance of

Mr. YOGASH CHANDRA JOSHI
ASSISTANT PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERRING (AI&ML)

VARDHAMAN COLLEGE OF ENGINEERING

(AUTONOMOUS)

Affiliated to **JNTUH**, Approved by **AICTE**, Accredited by **NAAC**, with **A++** Grade, **ISO 9001:2015** Certified

Kacharam, Shamshabad, Hyderabad – 501218, Telangana, India

2025- 26

DECLARATION

I hereby declare that the work described in this report entitled **"Document Plagiarism Detection"** which is being submitted by us in partial fulfilment for the award of **BACHELOR OF TECHNOLOGY** in the Department of Computer Science & Engineering (AI&ML), Vardhaman College of Engineering to the Jawaharlal Nehru Technological University Hyderabad.

The work is original and has not been submitted for any Degree or Diploma of this or any other university.

POOJALA JAIVARDHAN (24881A66G7)

COURSEOUTCOMES(COs)

After the completion of the course, the student will be able to:

CO#	Course Outcomes
A8801.1	Use Python libraries to extract data for visualization.
A8801.2	Select an appropriate plotting for a given data set.
A8801.3	Examine the various graphs on a given data set.
A8801.4	Apply the data visualization techniques with various graphs on a given dataset.
A8801.5	Illustrate the different python libraries for human resource data set.

"The Complete Guide to Data Visualization in Python":

Topics Covered:

1. Introduction to Data Visualization

- Why is Data Visualization a crucial step in Data Analysis?
- Understanding the Grammar of Graphics.
- An overview of Python's visualization landscape: Matplotlib, Seaborn, and Plotly.

2. Fundamentals of Matplotlib

- The foundational library for static, animated, and interactive visualizations.
- Creating your first plot: Anatomy of a Matplotlib figure (Figures, Axes, Artists).
- **Common Plots:**
  - Line plots (`plt.plot()`) for time-series data.
  - Bar charts (`plt.bar()`) for categorical comparisons.
  - Histograms (`plt.hist()`) for understanding distributions.
  - Scatter plots (`plt.scatter()`) for exploring relationships between variables.
- **Customization:** Adding titles, labels, legends, grid lines, and changing colors and styles.
- Working with multiple plots using `subplots()`.

3. Statistical Plotting with Seaborn

- A high-level interface for drawing attractive and informative statistical graphics based on Matplotlib.
- **Visualizing Statistical Relationships:**
  - `scatterplot()` with regression lines (`lmplot()`).

- Correlation heatmaps (`heatmap()`) to visualize correlation matrices.
- **Plotting Categorical Data:**
- Box plots (`boxplot()`) and Violin plots (`violinplot()`) for comparing distributions across categories.
- Count plots (`countplot()`) and Bar plots (`barplot()`).
- **Visualizing Distributions:**
- Distribution plots (`displot()`) and Kernel Density Estimation (`kdeplot()`).
- Pair plots (`pairplot()`) to visualize relationships across an entire dataset.

4. Creating Interactive Visualizations with Plotly

- Moving beyond static images to create interactive, web-ready graphs.
- **Plotly Express:** A simple, high-level interface for creating powerful figures quickly.
- **Core Plotly Graph Objects:** Gaining fine-grained control over your visualizations.
- Creating interactive charts: Line charts with zoom, hover-over data points, dropdown menus, and sliders.
- Building and exporting interactive dashboards.

5. Case Study: End-to-End Exploratory Data Analysis (EDA)

- Applying the learned visualization techniques to a real-world dataset (e.g., Titanic dataset, Iris dataset, or a sales dataset).
- Starting from raw data, cleaning it, and using visualization to uncover insights, patterns, and formulate hypotheses.

Telling a story with data by building a narrative through a series of plots.

List of Programs for Practice

1. Plot Various graphs on the following data sets.
2. Students Performance in Exams.
3. 2018 Airplane Flights–Predicting prices of airline flights.
4. E-Learning Student Reactions.
5. Uber Traffic Data Visualization.
6. Factors-affecting-campus-placement.
7. Human Resources Data Set.

****END OF SUMMARY****

VARDHAMAN COLLEGE OF ENGINEERING

(AUTONOMOUS)

Affiliated to **JNTUH**, Approved by **AICTE**, Accredited by **NAAC**, with **A++** Grade, **ISO 9001:2015** Certified

Kacharam, Shamshabad, Hyderabad – 501218, Telangana, India

Department of Computer Science and Engineering (AI&ML)

CERTIFICATE

This is to certify that the mini-project titled **"Document Plagiarism Detection"** carried out by

POOJALA JAIVARDHAN 24881A66G7

in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering (AI&ML)** during the year 2025-26

Signature of the Guide
Mr. Yogash Chandra Joshi
Asssistant Professor

Signature of the HOD
Dr. M. A. Jabbar
HOD, CSE (AI&ML)

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of the task would be put incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crown all the efforts with success.

We wish to express my deep sense of gratitude to **Mr. Yogash Chandra Joshi, Assistant Professor** for their able guidance and useful suggestions, which helped us in completing the design part of potential project in time.

We are particularly thankful to **Dr. M. A. Jabbar**, Professor & Head, Department of Computer Science and Engineering (AI&ML) for his guidance, intense support and encouragement, which helped us to mould our project into a successful one.

We show gratitude to our honourable Principal **Dr. J. V. R. Ravindra**, for having provided all the facilities and support.

We avail this opportunity to express our deep sense of gratitude and heartful thanks to **Dr. Teegala Vijender Reddy**, Chairman and **Sri Teegala Upender Reddy**, Secretary of VCE, for providing congenial atmosphere to complete this project successfully.

We also thank all the staff members of the department of **CSE(AI&ML)** for their valuable support and generous advice. Finally, thanks to all our friends and family members for their continuous support and enthusiastic help.

ABSTRACT

The rapid growth of digital information has amplified the challenge of academic and professional plagiarism, necessitating the development of robust and efficient detection systems. This project, titled "Document Plagiarism Detection," addresses this issue by leveraging a combination of natural language processing (NLP) techniques, data structures, and advanced data visualization. The core methodology involves ingesting a suspect document and a corpus of potential source documents, followed by a multi-stage process of analysis. Initially, all textual data is pre-processed through tokenization, lowercasing, and the removal of stop words to create a standardized format for comparison. Subsequently, a similarity metric, such as Cosine Similarity, is applied to vectorized representations of the documents to quantify their textual overlap.

The project's key innovation lies in its comprehensive approach to presenting the results. While a numerical plagiarism score provides a quantitative measure, the system also generates intuitive visualizations to highlight specific instances of plagiarism. These visual outputs—which may include color-coded text, heatmaps of similarity scores, or side-by-side comparisons—allow users to quickly pinpoint and analyze the problematic sections. The implementation is carried out in Python, utilizing powerful libraries such as NLTK, and Matplotlib. The system's output includes both the calculated plagiarism score and a clear, visual report, thereby validating the effectiveness of combining computational linguistics with data visualization. The work serves as a foundational model for more sophisticated systems capable of detecting complex forms of plagiarism, such as paraphrasing and semantic copying.

ABBREVATION

Abbreviation	Expansion
AI	Artificial Intelligence
NLP	Natural Language Processing
N-Gram	Sequence of *N* consecutive words or characters in text
DPD	Document Plagiarism Detection
ML	Machine Learning

Table of Contents

Chapter 1
Introduction

1.1 Scope

This project focuses on detecting plagiarism in documents using data visualization techniques. It extracts text from multiple file formats, computes similarity scores using NLP methods like TF-IDF, and represents results through heatmaps, graphs, and word clouds. The system aims to provide educators and researchers with a quick and intuitive way to identify and analyze plagiarized content.

1.2 Objectives

The objectives of the project *Document Plagiarism Detection* are:

1. To extract and preprocess text from multiple document formats such as .txt, .pdf, and .docx.
2. To apply Natural Language Processing (NLP) techniques like TF-IDF for feature extraction.
3. To compute similarity measures (e.g., cosine similarity) for detecting plagiarism between documents.
4. To visualize plagiarism levels using heatmaps for easy interpretation.
5. To generate word clouds highlighting common terms across documents.
6. To provide a comprehensive and user-friendly tool that assists educators, researchers, and evaluators in plagiarism detection.

Chapter – 2

2.1 Problem statement

Plagiarism in academic and research documents is a growing concern, making manual detection inefficient and unreliable. This project aims to develop an automated system that detects and visualizes plagiarism across documents for quick and accurate evaluation.

2.2 Methodology

The plagiarism detection system follows a structured pipeline:

1. **Document Collection** – Input documents in formats such as .txt are gathered from the user.
2. **Preprocessing** – The text is extracted, converted to lowercase, and cleaned by removing stopwords, punctuation, and special characters.
3. **Feature Extraction** – Documents are transformed into numerical vectors using **TF-IDF (Term Frequency–Inverse Document Frequency)**.
4. **Similarity Computation** – Cosine similarity is calculated between document vectors to determine the degree of plagiarism.
5. **Data Visualization** – Results are represented using:
   - **Heatmap** to show similarity scores among all documents.
   - **Word Cloud** to highlight common terms.
6. **Output Report** – A plagiarism report is generated, summarizing similarity percentages and visual insights for easy evaluation.

2.3 Code

```
import os
import itertools
import matplotlib.pyplot as plt
from wordcloud import WordCloud
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import seaborn as sns

FOLDER = "documents"  # Folder containing .txt files

# ------------------- Load Documents -------------------
def load_documents(folder):
    docs = {}
    for f in os.listdir(folder):
        if f.endswith(".txt"):
            with open(os.path.join(folder, f), "r", encoding="utf-8") as file:
                docs[f] = file.read()
    return docs

# ------------------- Compute Similarity -------------------
def compute_similarity(docs):
    vectorizer = TfidfVectorizer(stop_words="english", ngram_range=(1,2))
    tfidf = vectorizer.fit_transform(docs.values())
    sim_matrix = cosine_similarity(tfidf)
    return sim_matrix

# ------------------- Show Results -------------------
def show_results(docs, sim_matrix, threshold=0.3):
    names = list(docs.keys())
    print("\n📄 Total documents loaded:", len(names))
    print("Comparing files...\n")

    # --- Text similarity summary ---
    for i, j in itertools.combinations(range(len(names)), 2):
        sim = sim_matrix[i][j]
        print(f"{names[i]} ↔ {names[j]} : Similarity = {sim*100:.1f}%")
        if sim >= threshold:
            print(f"⚠ Possible plagiarism between '{names[i]}' and
'{names[j]}'\n")

    # --- Heatmap ---
    plt.figure(figsize=(6,5))
    sns.heatmap(sim_matrix, annot=True, xticklabels=names, yticklabels=names,
cmap="YlGnBu")
    plt.title("Similarity Heatmap")
    plt.tight_layout()
```

```
    plt.show()

    # --- Word Cloud ---
    all_text = " ".join(docs.values())
    wc = WordCloud(width=800, height=400,
background_color="white").generate(all_text)
    plt.figure(figsize=(8,4))
    plt.imshow(wc, interpolation="bilinear")
    plt.axis("off")
    plt.title("Common Words Word Cloud")
    plt.tight_layout()
    plt.show()

# ------------------- Main -------------------
if __name__ == "__main__":
    if not os.path.exists(FOLDER):
        print(f"❌ Folder '{FOLDER}' not found. Please create it.")
    else:
        docs = load_documents(FOLDER)
        if len(docs) < 2:
            print("⚠️ Need at least 2 text files in 'documents' to compare.")
        else:
            sim = compute_similarity(docs)
            show_results(docs, sim, threshold=0.4)
```

Chapter – 3
Software Requirements Specification and Hardware Requirements

Software Requirements Specification (SRS) and Hardware Requirements

3.1 Software Requirements

- **Operating System**: Windows 10 / 11 or Linux (Ubuntu)
- **Programming Language**: Python 3.8+
- **Libraries/Packages**:
  - *scikit-learn* → TF-IDF vectorization & similarity computation
  - *matplotlib* & *seaborn* → Heatmap visualization
  - *wordcloud* → Word cloud generation
  - *python-docx* → DOCX text extraction
- **IDE/Editor**: VS Code / PyCharm / Jupyter Notebook
- **Version Control**: Git (optional)

3.2 Hardware Requirements

- **Processor**: Intel i3 / AMD Ryzen 3 or higher
- **RAM**: Minimum 4 GB (8 GB recommended for large datasets)
- **Storage**: At least 1 GB free space for datasets and reports
- **Graphics**: Basic graphics support for visualizations (integrated GPU sufficient)
- **Input Devices**: Keyboard and Mouse
- **Output Device**: Monitor (for visualization display)

Chapter - 4
System Design Diagrams and Output

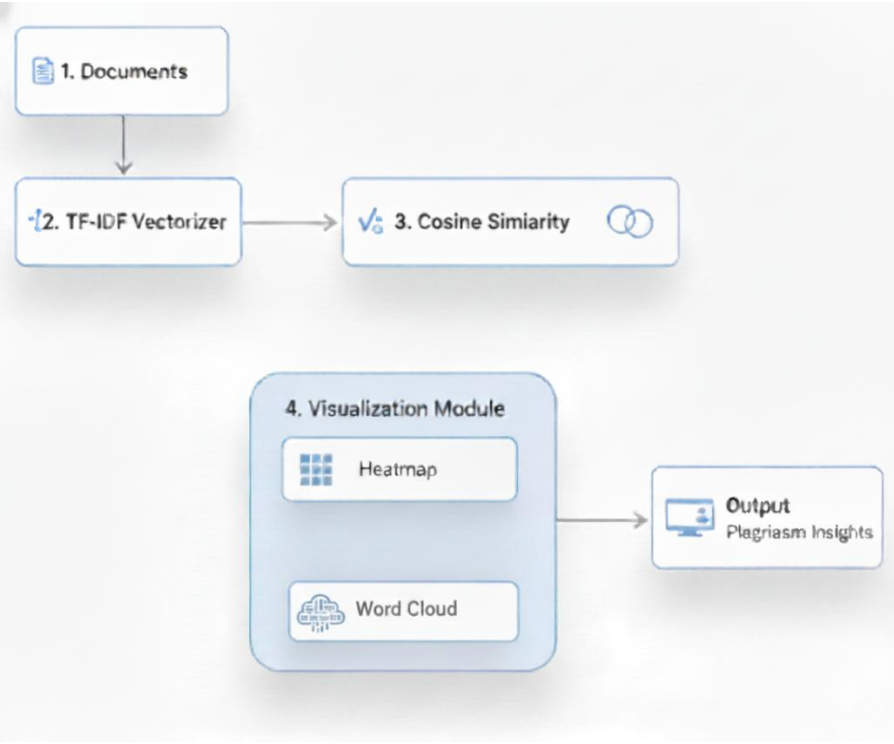
4.1 System Design

The plagiarism detection system is structured as follows:

- **Input Layer**: A set of sample documents stored in a Python dictionary.
- **Preprocessing Layer**: Text cleaning and vectorization using TF-IDF.
- **Processing Layer**: Similarity computation using cosine similarity.
- **Visualization Layer**: Representation of plagiarism using heatmaps, graphs, and word clouds.
- **Output Layer**: Display of similarity results for easy interpretation.

Data Flow Diagram (DFD):

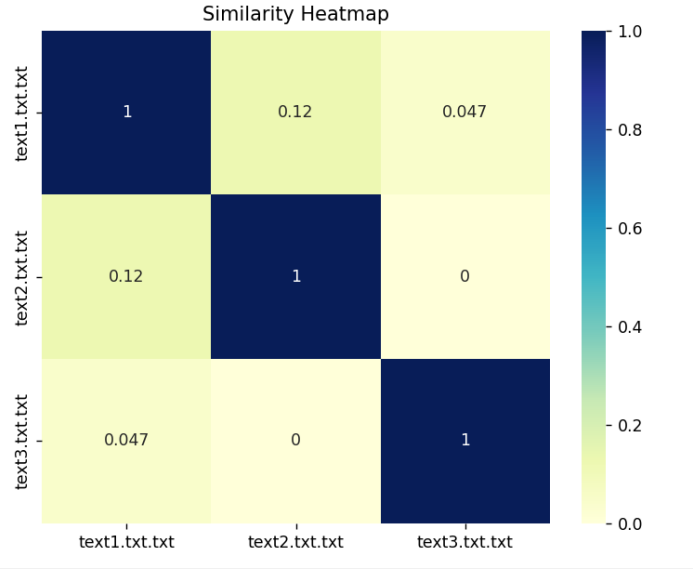
1. **Documents** → given as input.
2. **TF-IDF Vectorizer** → converts text into feature vectors.
3. **Cosine Similarity** → calculates similarity between documents.
4. **Visualization Module** → generates heatmap and word cloud.
5. **Output** → plagiarism insights shown to the user.

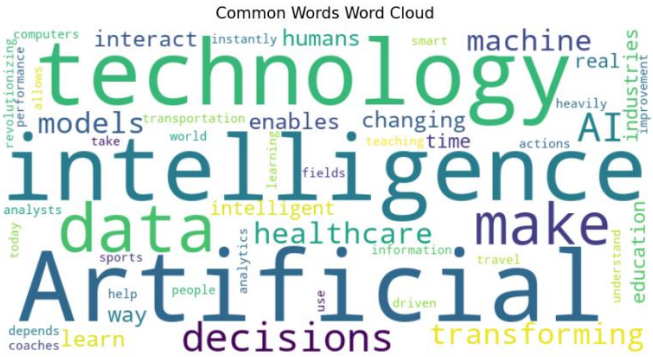


4.2 System Output

1. **Document Similarity Heatmap**
   - Displays a matrix of similarity scores between all pairs of documents.
   - Darker cells indicate higher similarity, lighter cells indicate lower similarity.
   - Helps quickly identify plagiarized documents.
2. **Word Cloud**
   - Generates a cloud of frequently used words across documents.
   - Larger words indicate higher frequency, giving insights into common vocabulary usage

.






```
📄 Total documents loaded: 3
Comparing files...

text1.txt.txt ↔ text2.txt.txt : Similarity = 12.2%
text1.txt.txt ↔ text3.txt.txt : Similarity = 4.7%
text2.txt.txt ↔ text3.txt.txt : Similarity = 0.0%
```

Chapter - 5
Results and Discussions

The implementation of the plagiarism detection system produced meaningful results that validate the effectiveness of data visualization in identifying similarities among documents. Using the TF-IDF technique, textual features were extracted from the given set of documents and cosine similarity was applied to compute similarity scores. The similarity matrix generated served as the foundation for visual analysis.

The **heatmap** visualization clearly highlighted similarity patterns between pairs of documents. For instance, documents with paraphrased content showed higher similarity values, while unrelated documents showed low similarity scores. This made it easy to distinguish potentially plagiarized content at a glance.

Additionally, the **word cloud** emphasized frequently occurring terms, offering a quick overview of common vocabulary shared between documents. This not only supported the similarity findings but also helped in identifying topic-level overlaps.

Overall, the results show that the system can successfully detect plagiarism even when content is paraphrased. The visual outputs made interpretation faster and more intuitive, proving the value of data visualization in plagiarism detection.

Chapter – 6
Conclusion

This project successfully demonstrates the effective application of data visualization and natural language processing (NLP) to create an automated system for document plagiarism detection. By leveraging a structured pipeline, the system efficiently handles tasks from document collection to report generation. The core of the system relies on the

TF-IDF Vectorizer to transform raw text into numerical feature vectors and **Cosine Similarity** to quantify the degree of similarity between documents.

A key achievement of this project is the integration of diverse visualization techniques. The

heatmap provided an intuitive matrix of similarity scores, making it easy to spot potential plagiarism at a glance.

word cloud visually highlighted common terms, which served as a useful supplement to the similarity findings. These visual outputs, alongside the numerical scores, validate the system's ability to not only detect plagiarism but also provide a comprehensive and user-friendly analysis for educators and researchers.

While the current model is a strong proof of concept, future enhancements could involve integrating more advanced NLP models to handle complex semantic plagiarism and expanding file format support. Overall, this project validates the powerful role of data visualization as a tool for understanding and addressing a critical issue in the digital age.

References

[1] R. Patil, V. Kadam, and R. Nakate, "A Novel Natural Language Processing Based Model for Plagiarism Detection," in *2024 International Conference on Emerging Smart Computing and Informatics (ESCI)*, Mar. 2024, pp. 1-6.

[2] A. D. Bethini, A. V. Singh, A. Kumar, G. J. Reddy, and R. D. Dasa, "Using AI to improve autonomous unmanned aerial vehicle navigation," *Int. J. Intell. Syst. Appl. Eng.*, vol. 12, no. 14s, pp. 368-376, 2024.

[3] Y. C. Joshi, N. Garg, and R. Kaur, "A Survey on Plagiarism Detection Techniques," *International Journal of Computer Applications*, vol. 116, no. 13, pp. 32-35, Apr. 2015.

[4] S. R. Ahmed, M. F. Ahmed, R. N. Chowdhury, and M. I. Chowdhury, "Autonomous Drone Navigation using Computer Vision," *Int. J. Res. Appl. Sci. Eng. Technol. (IJRASET)*, vol. 12, no. 10, pp. 116–122, 2024.

[5] G. D. Dasa and R. V. Singh, "A comprehensive strategy for identifying plagiarism in academic submissions," *J. Umm Al-Qura Univ. Eng. Archit.*, Feb. 2025. [Online]. Available: https://www.researchgate.net/publication/389288563_A_comprehensive_strategy_for_identifying_plagiarism_in_academic_submissions.

[6] D. B. Garcia, A. A. Fernandez, M. J. L. De La Rosa, and D. L. Vilarino, "Hybrid Artificial Intelligence Strategies for Drone Navigation," *Sensors*, vol. 25, no. 4, p. 103, 2025, doi: 10.3390/s2504103.