Mid Semester Presentation Project-Plagiarism Detector

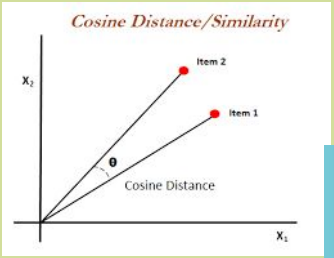
Supervised by:Dr. Alekha Kumar Mishra

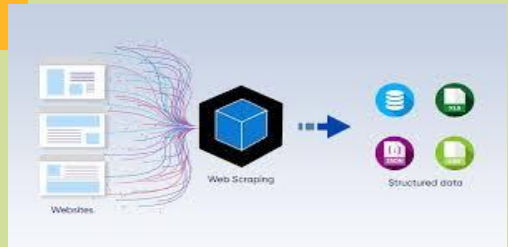
Presented by:Nidhi Dubey
(2022PGCSCA074)

Table Of Contents

INTRODUCTION

- A tool to detect plagiarism in text-based files.
- Uses machine learning (TF-IDF + Cosine Similarity) and web scraping.
- Supports multiple file types (PDF, DOCX, TXT, Images).
- Provides color-coded similarity reports.





File / Source	Similarity (%)
OOP_Java_Class_Objects_Setters_Getters_Constructors.docx	100.0%
OS_Full_Notes.pdf	21.34%
https://en.wikipedia.org/wiki/Natural_language_processing	21.33%
https://developer.mozilla.org/en-US/docs/Web/JavaScript	19.51%
https://stackoverflow.com/questions/tagged/python	17.18%
https://www.geeksforgeeks.org/python-programming-language/	13.94%
https://www.bbc.com/news	12.24%
https://docs.oracle.com/en/java/	11.87%
https://hackernoon.com/	11.07%
Unix_Linux_Servers.docx	11.04%
https://pubmed.ncbi.nlm.nih.gov/	10.58%
https://flask.palletsprojects.com/en/2.2.x/	10.43%
https://www.tensorflow.org/	10.11%
https://arxiv.org/	7.16%
sample.txt	6.58%

Objectives:-

- Automate plagiarism detection process.
- Provide accurate similarity scores in percentage using cosine similarity algorithm.
- Compare against both stored files and online sources.
- Improve accuracy using OCR and better text extraction.
- Web interface for easy file upload and result visualization

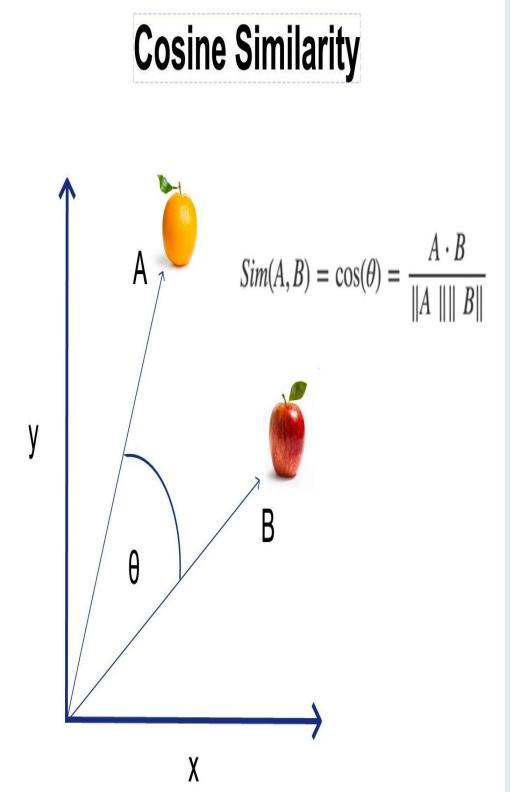
Cosine Similarity Algorithm

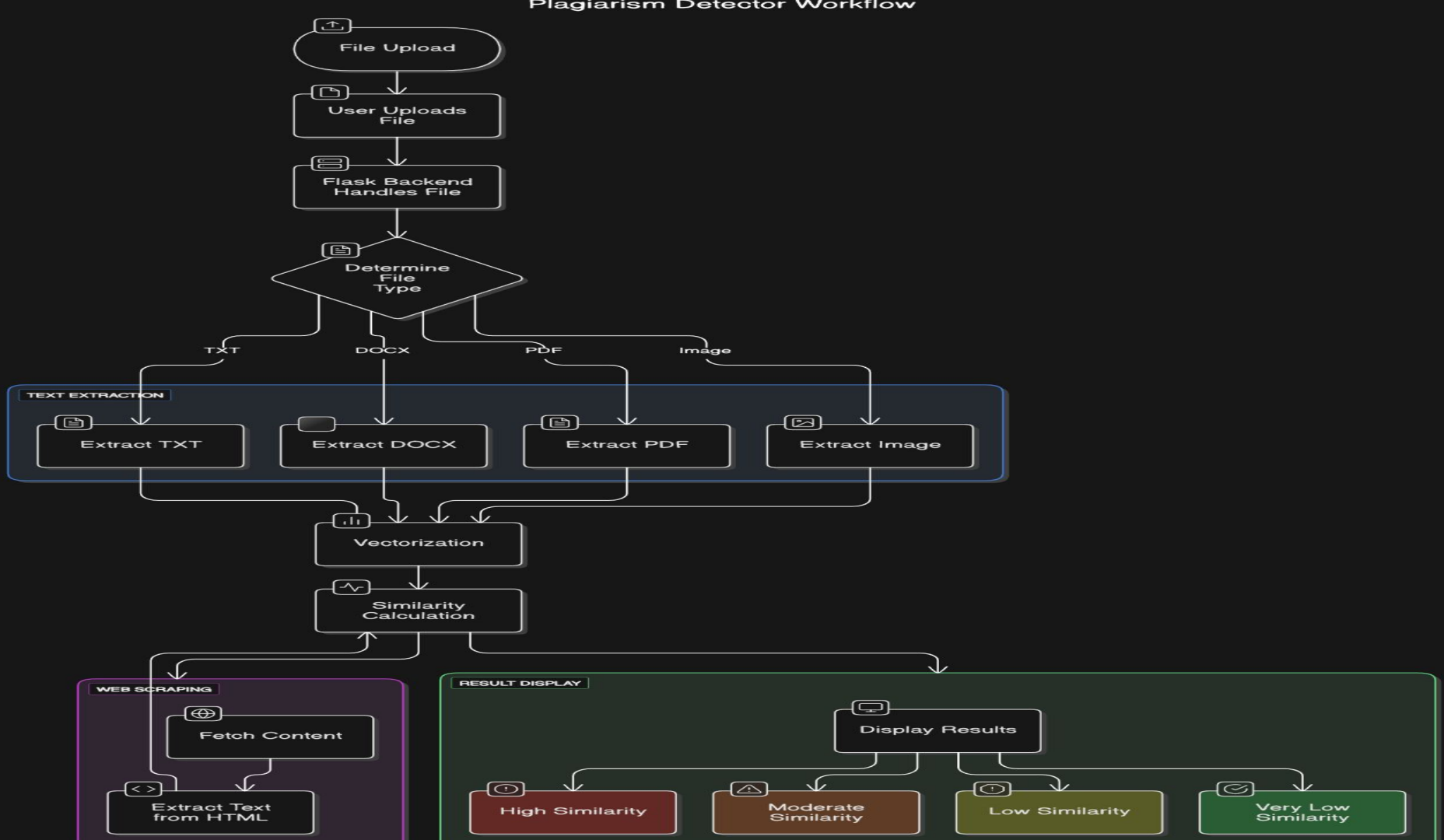
Steps and Results

Cosine Similarity measures the cosine of the angle between two vectors in a multi-dimensional space

- Steps:
  1. Convert text into numerical vectors using TF-IDF.
  2. Compute the dot product of the vectors.
  3. Divide by the product of their magnitudes.

- Result: A similarity score ranging from 0 (completely different) to 1 (identical)



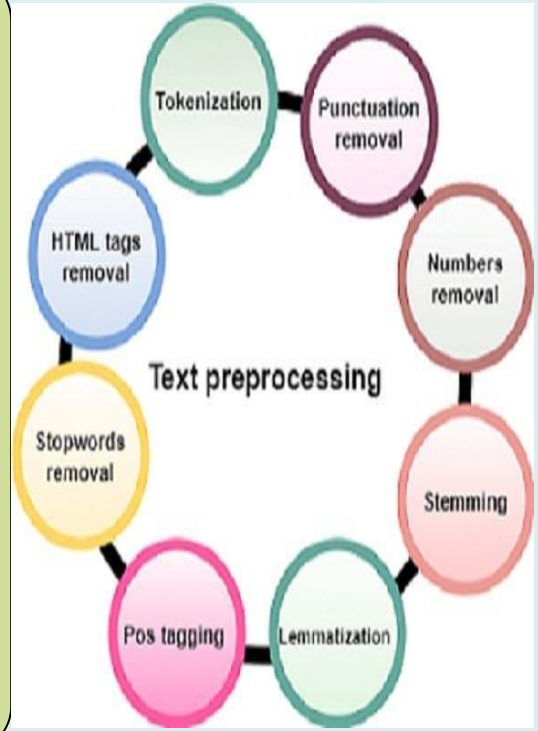

Plagiarism Detector Workflow
File Upload
User Uploads File
Flask Backend Handles File
Determine File Type
TXT
DOCX
PDF
Image
TEXT EXTRACTION
Extract TXT
Extract DOCX
Extract PDF
Extract Image
Vectorization
Similarity Calculation
WEB SCRAPING
Fetch Content
Extract Text from HTML
RESULT DISPLAY
Display Results
High Similarity
Moderate Similarity
Low Similarity
Very Low Similarity

Tokenization and Preprocessing

An Example

- Sentence A: 'The quick brown fox jumps over the lazy dog.'
- Sentence B: 'The fast brown fox leaps over a sleepy dog.'
- Tokens after preprocessing:

Sentence A: ['quick', 'brown', 'fox', 'jumps', 'lazy', 'dog']
Sentence B: ['fast', 'brown', 'fox', 'leaps', 'sleepy', 'dog']



Vectorization

Vector and Tf reprsentation

- Combined Vocabulary:-
  ['quick', 'fast', 'brown', 'fox', 'jumps', 'leaps', 'lazy', 'sleepy', 'dog']

- Vector Representation:-

Sentence A: [1, 0, 1, 1, 1, 0, 1, 0, 1]
Sentence B: [0, 1, 1, 1, 0, 1, 0, 1, 1]

Term	Sentence A (TF)	Sentence B (TF)
quick	1	0
fast	0	1
brown	1	1
fox	1	1
jumps	1	0
leaps	0	1
lazy	1	0
sleepy	0	1
dog	1	1

Section 1

Cosine Similarity Calculation

Result

- Cosine Similarity Formula:

Cosine Similarity = (Dot Product of Vectors) / (Magnitude of Vector A × Magnitude of Vector B)
Dot Product: 3
Magnitude of A: $\sqrt{6}$
Magnitude of B: $\sqrt{6}$

- Final Result:

Cosine Similarity = 3 / ($\sqrt{6}$ × $\sqrt{6}$) = 0.5 (50%)

Section 1

Technologies Used:-

Tech Stack

- **Frontend:** HTML, CSS, JavaScript
- **Backend:** Python (Flask)
- **Libraries:**
    i. scikit-learn (TF-IDF, Cosine Similarity)
    ii. BeautifulSoup + aiohttp – Web scraping
    iii. PyMuPDF + pytesseract – PDF and OCR handling
- **Algorithms:** Cosine Similarity

result

Result Display

- Displays similarity percentage in a table.

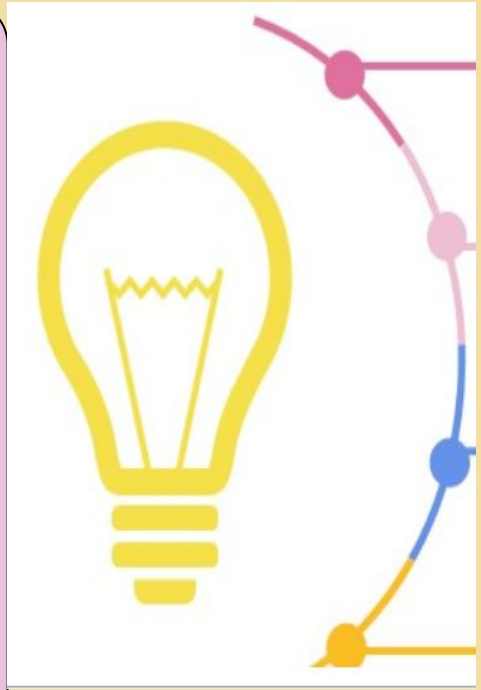
Color Codes:

- 🔴 High Similarity → ≥80%
- 🟠 Moderate Similarity → ≥50%
- 🟡 Low Similarity → ≥20%
- 🟢 Very Low Similarity → <20%

File / Source	Similarity (%)
OOP_Java_Class_Objects_Setters_Getters_Constructors.docx	100.0%
OS_Full_Notes.pdf	21.34%
https://en.wikipedia.org/wiki/Natural_language_processing	21.33%
https://developer.mozilla.org/en-US/docs/Web/JavaScript	19.51%
https://stackoverflow.com/questions/tagged/python	17.18%
https://www.geeksforgeeks.org/python-programming-language/	13.94%
https://www.bbc.com/news	12.24%
https://docs.oracle.com/en/java/	11.87%
https://hackernoon.com/	11.07%
Unix_Linux_Servers.docx	11.04%
https://pubmed.ncbi.nlm.nih.gov/	10.58%
https://flask.palletsprojects.com/en/2.2.x/	10.43%
https://www.tensorflow.org/	10.11%
https://arxiv.org/	7.16%
sample.txt	6.58%

Future Scope

- Add support for `.csv`, `.xlsx` files.
- Improve OCR accuracy using TensorFlow or a custom LSTM model.
- Allow users to set custom similarity thresholds.
- Add multi-language support.
- Improve short text similarity detection using context-aware embeddings (like BERT).



Conclusion

- Successfully developed a plagiarism detection system with high accuracy and reliable performance.
- Leveraged machine learning, web scraping, and OCR for robust text extraction and similarity calculation.
- Enhanced scalability and efficiency through async processing and improved text extraction techniques.
- Future improvements can further enhance accuracy and user experience.

THANK YOU