AMRITA VISHWA VIDYAPEETHAM

Plagiarism Detection using Automata

- Final End Sem Review -

Formal Language and Automata (FLA)

D Batch - 07

Malavika S Prasad - [CB.SC.U4AIE23315]

Vibhu Sanchana - [CB.SC.U4AIE23347]

Asmi K - [CB.SC.U4AIE23351]

Geshna B - [CB.SC.U4AIE23360]

Introduction

Plagiarism is widespread in academics, research, and online content. Detecting copied or near-duplicate text manually is inefficient.

To develop a Formal Automata–based AI Plagiarism Detection System that can:

Detect plagiarism or text similarity between user input and reference content, Work in a topic-specific manner by using Wikipedia or similar databases, Use formal language and automata theory for efficient pattern recognition.

Problem Statement

Main Problems:

- Checking documents is too slow with normal methods and we need to check against many sources at once
- Students copy-paste from multiple places
- Universities need fast, accurate detection

Technical Issues:

- Traditional search: $O(n \times m \times k)$ = very slow
- Multiple patterns = multiple scans = waste time

What We Solve:

- Fast multi-pattern search in one go
- Merge overlapping copied parts that calculates exact plagiarism percentage

Core Idea & Outcome

Core Idea

- Convert both input and reference text into sequences of tokens (words).
- Use pattern-matching automata to identify direct and partial matches.

Outcome: An intelligent plagiarism detector that is:

- Accurate (detects both literal and rephrased content)
- Fast (automata-based scanning)
- Adaptive (handles multiple writing styles)

Concepts

The system uses core ideas from Formal Language Theory and Automata to represent, store, and detect text patterns.

It bridges theoretical computer science and natural language processing.

Alphabets and Strings

1. Alphabets (Σ)
   - An alphabet is a set of characters or symbols used in text.
   - Example: Σ = {a–z, A–Z, 0–9, punctuation marks}.
   - Defines all possible symbols the automaton can read.
2. Strings
   - A string is a finite sequence of characters from Σ.
   - Example: “machinelearning” or “AI detection system”.
   - Each document or sentence is treated as a string over Σ.

Use in Project

- Both input and reference texts are converted into such strings.
- This allows the automaton to perform pattern matching formally and efficiently.

3. Language:

- A language is a set of strings over the same alphabet Σ.
- It represents a structured collection of valid text patterns.

Used for:

- Reference content (e.g., Wikipedia topic data) → forms Language L_ref.
- User's input text → a string s_input.
- The task → find whether s_input or its substrings exist within L_ref.

Formal Comparison

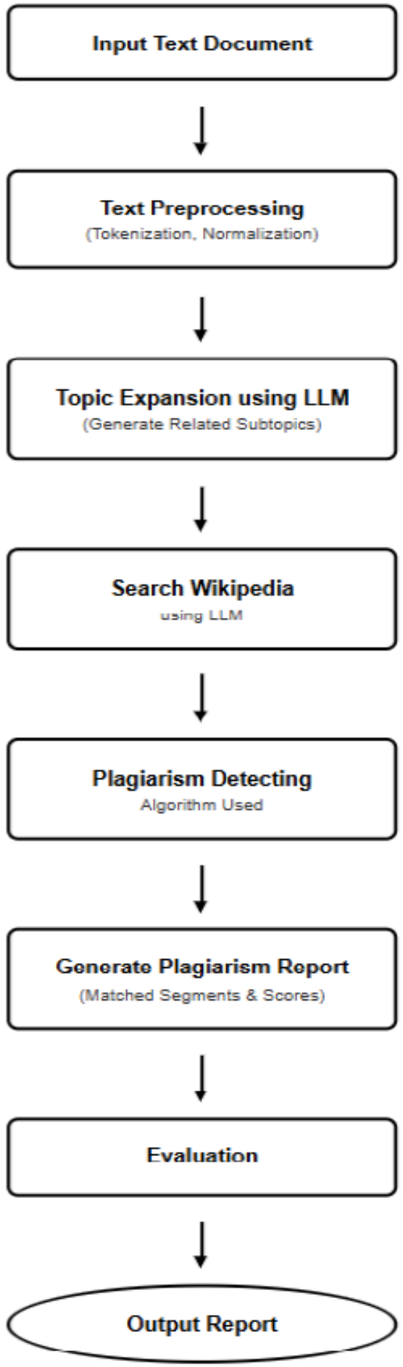
L_ref = {all reference sentences}

s_input ∈ L_ref ? → Plagiarism Detected

s_input ≈ L_ref ? → Semantic or Paraphrased Match

System Flow

- Input Text → Preprocessing
- Automaton-based Matching
- Topic Expansion (LLM)
- Context Retrieval
- Similarity Evaluation
- Generate Detailed Report



Workflow: From Input to Output

Step 1: User Inputs

- User provides:
  - Text to check (s_input)
  - Topic keyword (topic_name)
- Example: Topic → “Artificial Intelligence”

Step 2: Reference Data Collection

- LLM (Groq) fetches Wikipedia text related to the topic.
- Output → s_reference, a single structured document for comparison.

Step 3: Preprocessing

- Clean and prepare both texts:
- Convert to lowercase
- Remove punctuation
- Tokenize into words
- Remove common stopwords
- Purpose → ensure consistency in automata pattern matching.

Workflow: From Input to Output

Step 4: Build Automaton

- Construct a Trie from s_reference.
- Add failure links → forms the Aho–Corasick automaton.

Step 5: Pattern Matching

- Feed s_input into automaton.
- Record all matches and overlaps found by DFA logic.

Step 6: Compute Similarity

Provides a measurable plagiarism score.

$$\text{Similarity } \% = \frac{\text{Matched Tokens}}{\text{Total Tokens in Input}} \times 100$$

Step 7: Display Results

- Similarity Percentage
- Highlighted plagiarized phrases
- Color-coded match levels

DFA

What is a DFA?

- A Deterministic Finite Automaton (DFA) is a mathematical model used to recognize patterns in strings.
- Formal definition: DFA = $(Q, \Sigma, \delta, q_0, F)$
  - $Q \rightarrow$ Set of states
  - $\Sigma \rightarrow$ Input alphabet (symbols/characters)
  - $\delta \rightarrow$ Transition function (defines how states change for each input symbol)
  - $q_0 \rightarrow$ Start state
  - $F \rightarrow$ Set of accepting/final states

DFA Characteristics

- For each state and input symbol, the next state is unique → deterministic.
- Can accept or reject strings by reaching a final state.
- Can be represented as a state diagram (nodes = states, arrows = transitions).

Trie (Prefix Tree) Concept

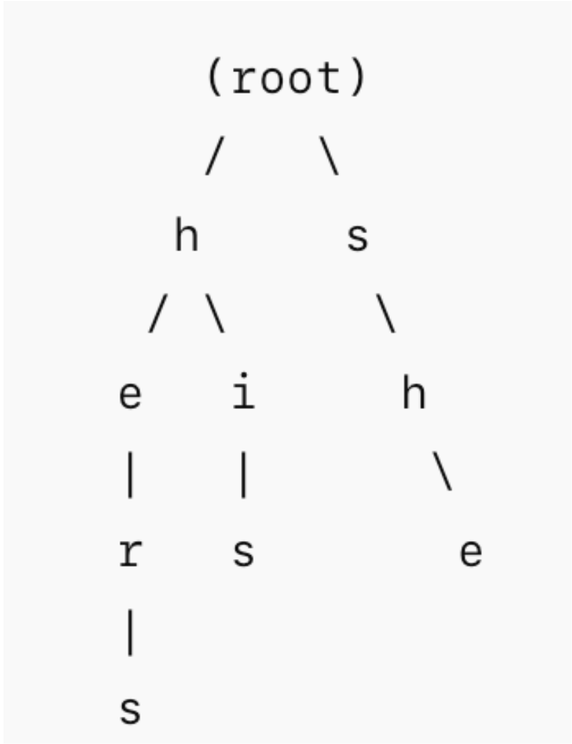
A Trie is a tree-based data structure used to store strings efficiently, especially when dealing with prefix-based search or pattern matching.

Key Concepts:

- Each node represents a character of a word.
- Each path from root to leaf represents a word.
- Common prefixes are stored only once, saving space.
- Time complexity: O(m) per word (m = length of word).

Why It's Used in Our Project:

- We use a Trie in the Aho-Corasick algorithm to store all possible patterns (substrings from user text).
- Enables simultaneous multi-pattern matching.
- Acts as the base structure of the automaton (each node = DFA state).



Insert words: "he", "she", "his", "hers"

Aho–Corasick Algorithm

- The Aho–Corasick algorithm is a highly efficient multi-pattern string searching algorithm.
- It builds a trie from a set of patterns like phrases or words to detect and augments it with failure links that allow the search to transition smoothly when mismatches occur.

Why we chose this:

- It enables the system to search for many reference phrases or copied segments at once, rather than scanning separately for each pattern, drastically improving speed.
- Add failure links that help jump between partial matches. These links avoid restarting the search from scratch on mismatches, allowing smooth transitions to the longest possible matching prefix, which reduces redundant comparisons.

1. Problem Context
   - Input text = string of tokens (words)
   - Reference text = set of valid phrases (language)
   - Goal = check if input string matches patterns in the reference language
2. Implementation Using DFA
   1. Trie Construction
      - Build a Trie with all reference words/phrases.
      - Each node = DFA state
      - Each edge = transition for a character/token
   2. Failure Links (Aho–Corasick)
      - Extend the Trie to handle overlapping patterns.
      - Acts as a DFA with additional transitions for mismatches.
      - Ensures linear-time multi-pattern search
   3. Traversal
      - Feed the user input text token by token.
      - DFA transitions from state to state according to input.
      - Whenever a final/accepting state is reached → a match is detected

Complexity and Efficiency

Time Complexity: O(n + m + z)

- n = length of input text
- m = total length of reference patterns
- z = number of matches found

Space Complexity: Depends on the total number of states and transitions in the automaton.

- Efficiency:

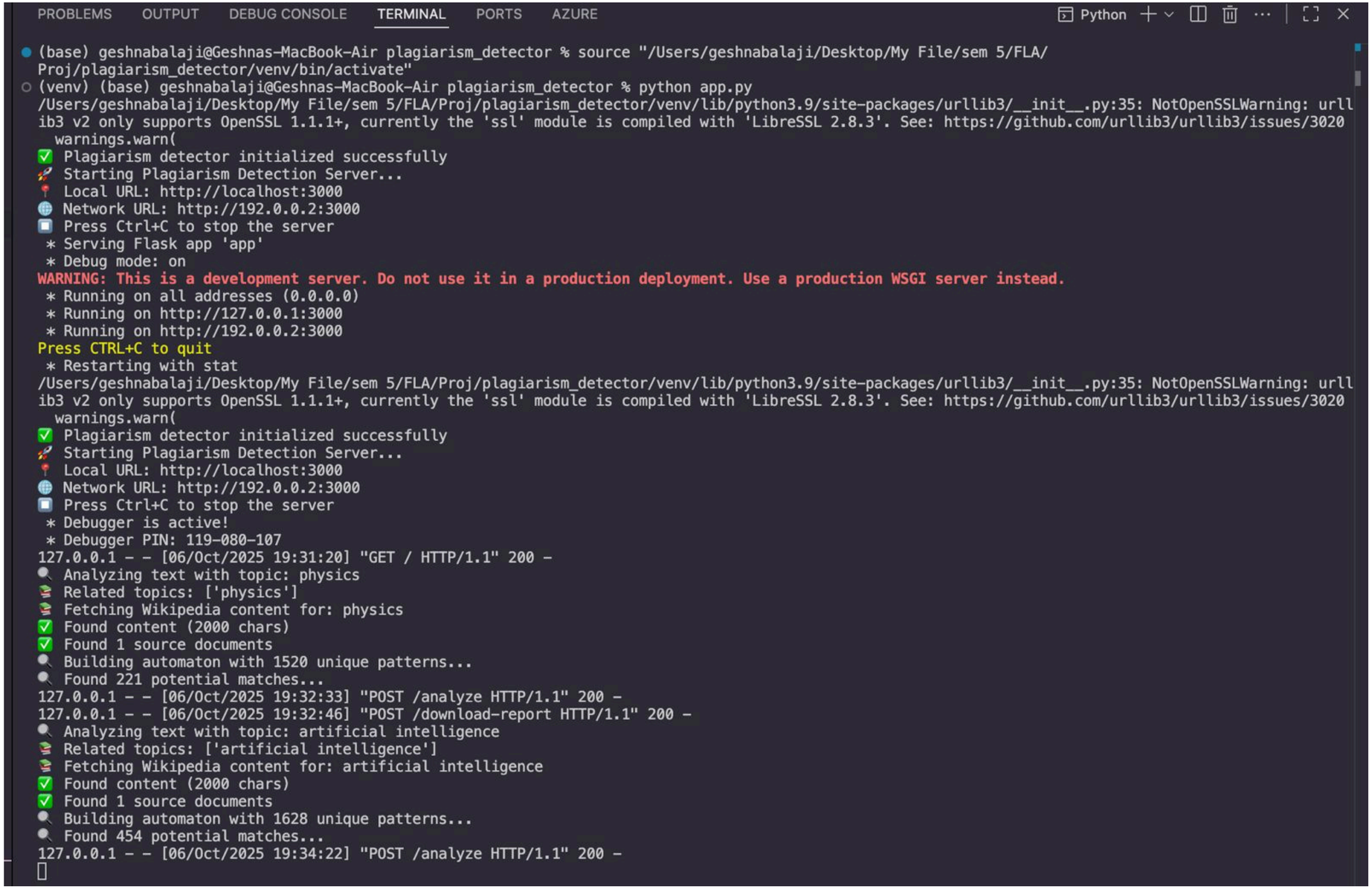
Performs pattern matching in real time even with large datasets.

Scales linearly → ideal for big academic repositories or corpora.

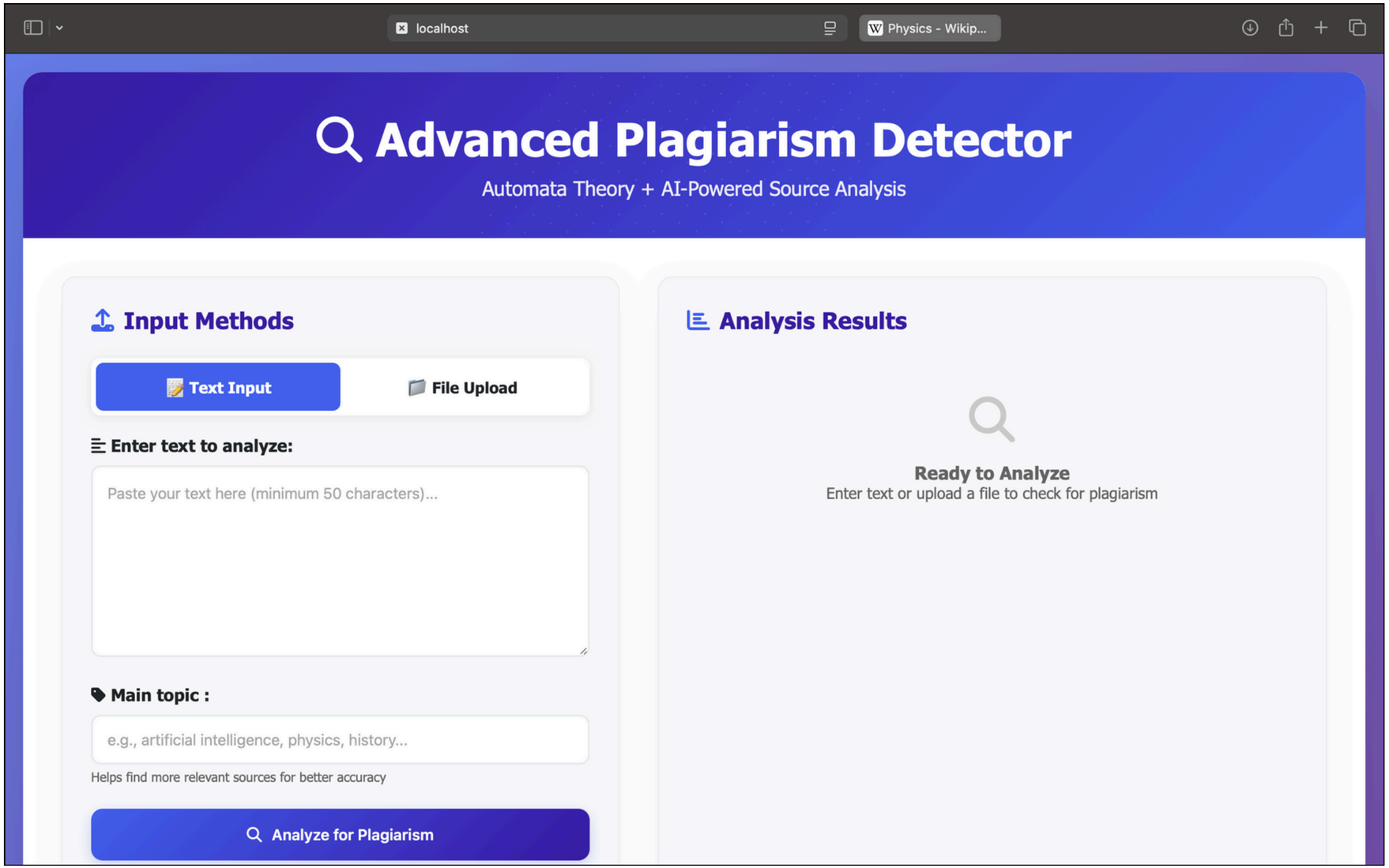
Advantages

- Works in linear time
- Detects overlapping and nested matches
- Ideal for large datasets like Wikipedia articles

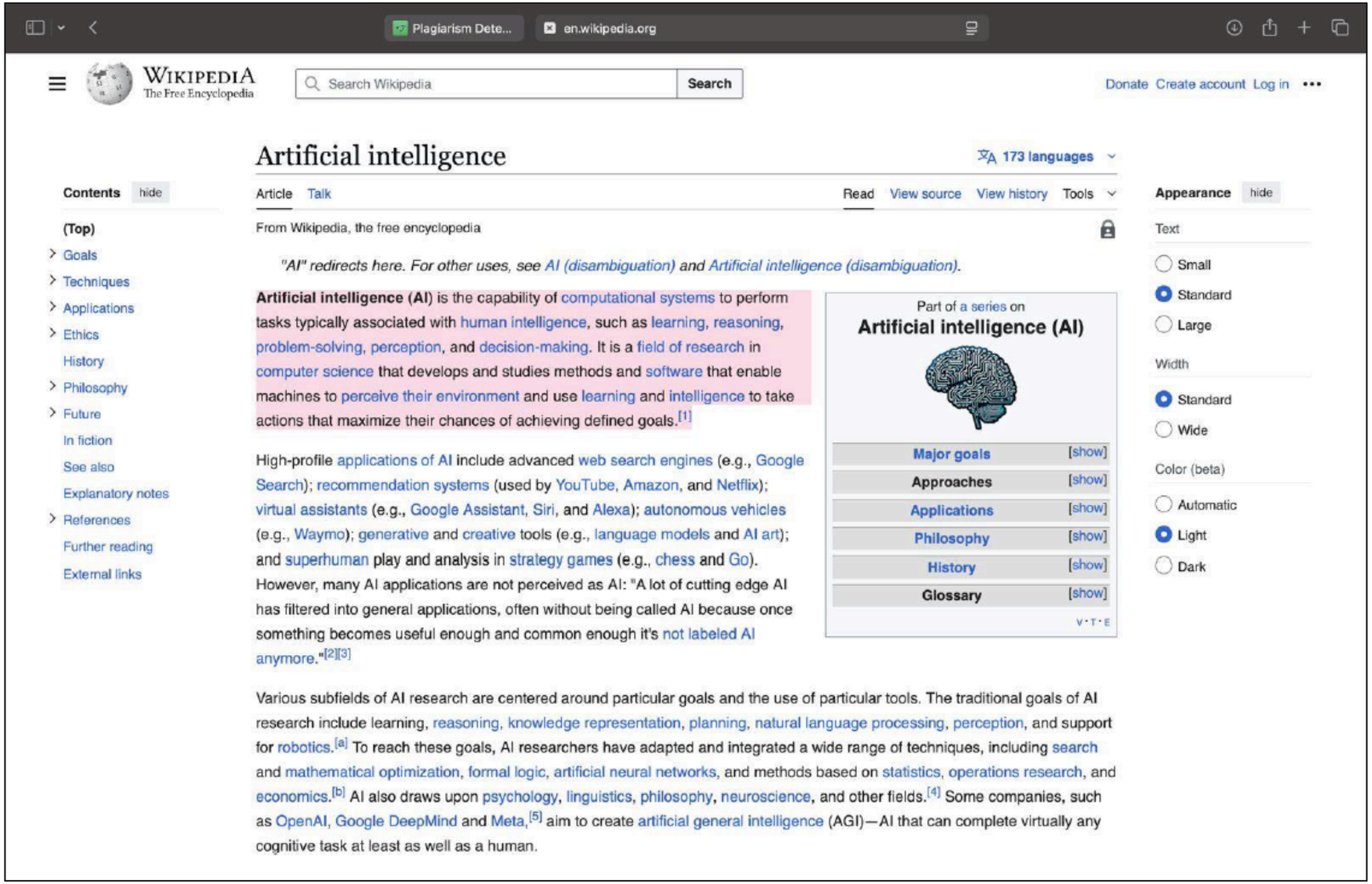
Results


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
AZURE
Python
(base) geshnabalaji@Geshnas-MacBook-Air plagiarism_detector % source "/Users/geshnabalaji/Desktop/My File/sem 5/FLA/Proj/plagiarism_detector/venv/bin/activate"
(venv) (base) geshnabalaji@Geshnas-MacBook-Air plagiarism_detector % python app.py
/Users/geshnabalaji/Desktop/My File/sem 5/FLA/Proj/plagiarism_detector/venv/lib/python3.9/site-packages/urllib3/__init__.py:35: NotOpenSSLWarning: urllib3 v2 only supports OpenSSL 1.1.1+, currently the 'ssl' module is compiled with 'LibreSSL 2.8.3'. See: https://github.com/urllib3/urllib3/issues/3020
warnings.warn(
✅ Plagiarism detector initialized successfully
🚀 Starting Plagiarism Detection Server...
📍 Local URL: http://localhost:3000
🌐 Network URL: http://192.0.0.2:3000
⏹️ Press Ctrl+C to stop the server
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:3000
* Running on http://192.0.0.2:3000
Press CTRL+C to quit
* Restarting with stat
/Users/geshnabalaji/Desktop/My File/sem 5/FLA/Proj/plagiarism_detector/venv/lib/python3.9/site-packages/urllib3/__init__.py:35: NotOpenSSLWarning: urllib3 v2 only supports OpenSSL 1.1.1+, currently the 'ssl' module is compiled with 'LibreSSL 2.8.3'. See: https://github.com/urllib3/urllib3/issues/3020
warnings.warn(
✅ Plagiarism detector initialized successfully
🚀 Starting Plagiarism Detection Server...
📍 Local URL: http://localhost:3000
🌐 Network URL: http://192.0.0.2:3000
⏹️ Press Ctrl+C to stop the server
* Debugger is active!
* Debugger PIN: 119-080-107
127.0.0.1 - - [06/Oct/2025 19:31:20] "GET / HTTP/1.1" 200 -
🔍 Analyzing text with topic: physics
📚 Related topics: ['physics']
📚 Fetching Wikipedia content for: physics
✅ Found content (2000 chars)
✅ Found 1 source documents
🔍 Building automaton with 1520 unique patterns...
🔍 Found 221 potential matches...
127.0.0.1 - - [06/Oct/2025 19:32:33] "POST /analyze HTTP/1.1" 200 -
127.0.0.1 - - [06/Oct/2025 19:32:46] "POST /download-report HTTP/1.1" 200 -
🔍 Analyzing text with topic: artificial intelligence
📚 Related topics: ['artificial intelligence']
📚 Fetching Wikipedia content for: artificial intelligence
✅ Found content (2000 chars)
✅ Found 1 source documents
🔍 Building automaton with 1628 unique patterns...
🔍 Found 454 potential matches...
127.0.0.1 - - [06/Oct/2025 19:34:22] "POST /analyze HTTP/1.1" 200 -

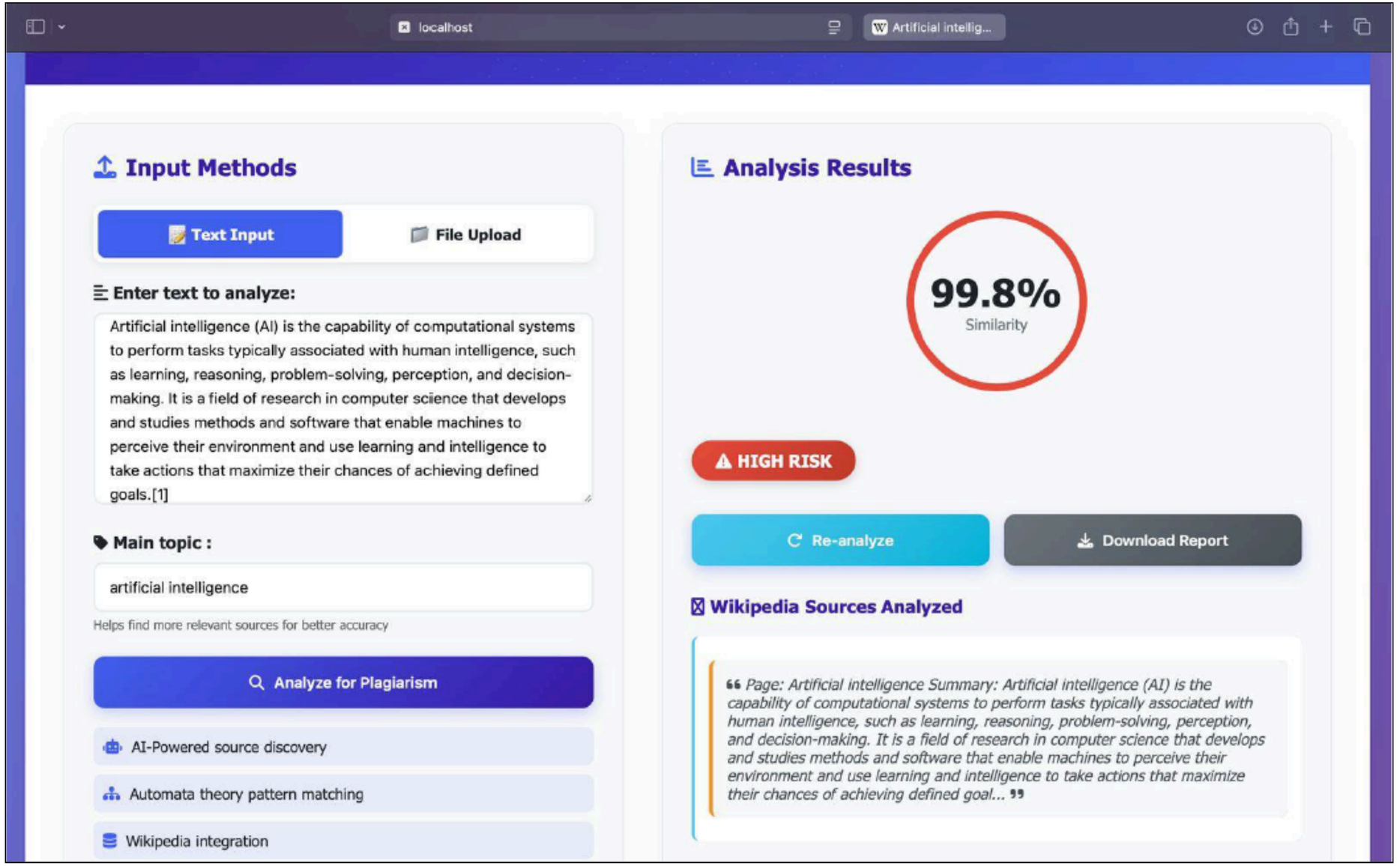
Results


localhost
Physics - Wikip...
Advanced Plagiarism Detector
Automata Theory + AI-Powered Source Analysis
Input Methods
Text Input
File Upload
Enter text to analyze:
Paste your text here (minimum 50 characters)...
Main topic :
e.g., artificial intelligence, physics, history...
Helps find more relevant sources for better accuracy
Analyze for Plagiarism
Analysis Results
Ready to Analyze
Enter text or upload a file to check for plagiarism

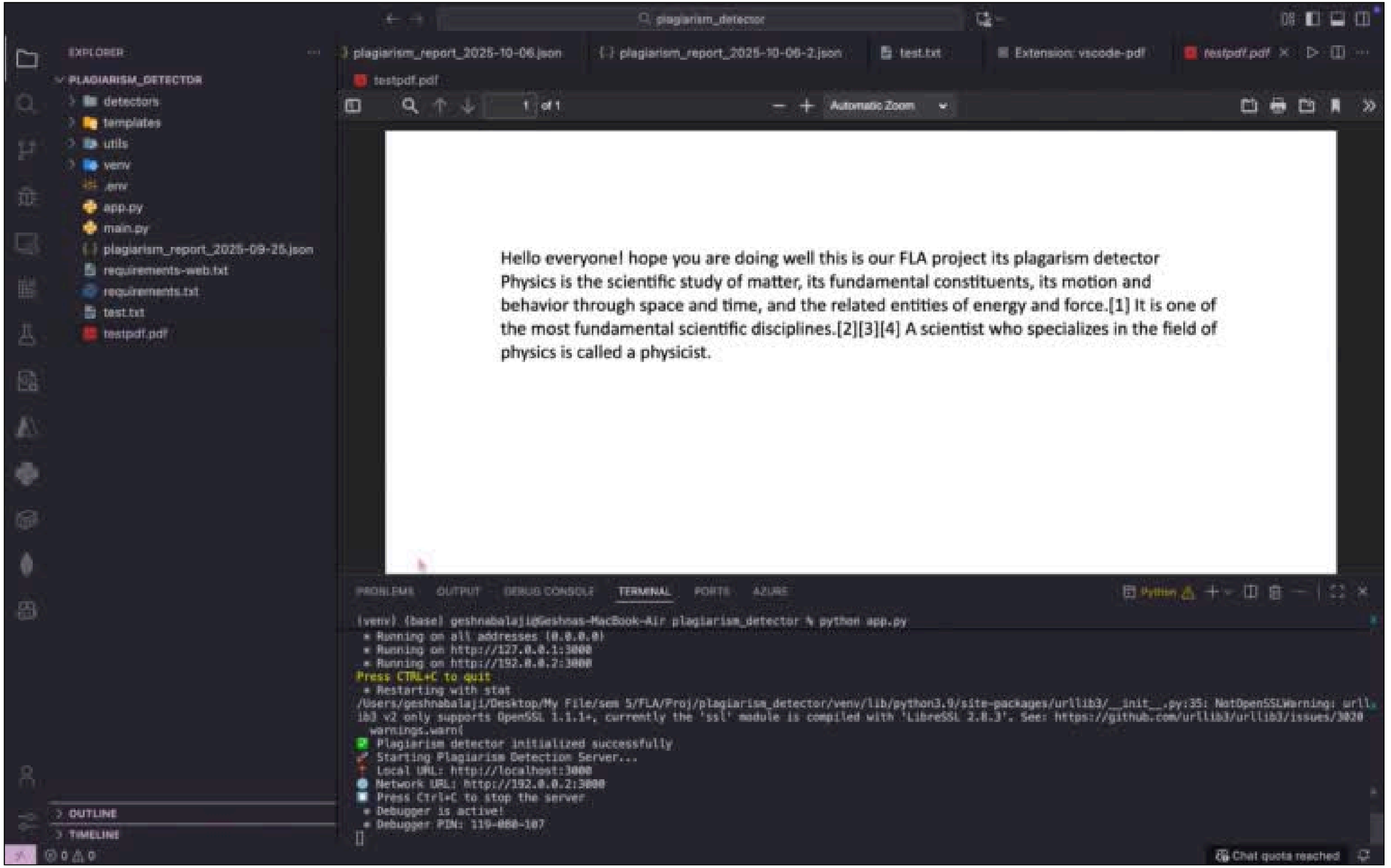
Results


Plagiarism Dete...
en.wikipedia.org
WIKIPEDIA
The Free Encyclopedia
Search Wikipedia
Search
Donate Create account Log in
Artificial intelligence
173 languages
Contents hide
(Top)
Goals
Techniques
Applications
Ethics
History
Philosophy
Future
In fiction
See also
Explanatory notes
References
Further reading
External links
Article Talk
Read View source View history Tools
Appearance hide
Text
Small
Standard
Large
Width
Standard
Wide
Color (beta)
Automatic
Light
Dark
From Wikipedia, the free encyclopedia
"AI" redirects here. For other uses, see AI (disambiguation) and Artificial intelligence (disambiguation).
Artificial intelligence (AI) is the capability of computational systems to perform tasks typically associated with human intelligence, such as learning, reasoning, problem-solving, perception, and decision-making. It is a field of research in computer science that develops and studies methods and software that enable machines to perceive their environment and use learning and intelligence to take actions that maximize their chances of achieving defined goals.[1]
Part of a series on
Artificial intelligence (AI)
Major goals [show]
Approaches [show]
Applications [show]
Philosophy [show]
History [show]
Glossary [show]
v · t · e
High-profile applications of AI include advanced web search engines (e.g., Google Search); recommendation systems (used by YouTube, Amazon, and Netflix); virtual assistants (e.g., Google Assistant, Siri, and Alexa); autonomous vehicles (e.g., Waymo); generative and creative tools (e.g., language models and AI art); and superhuman play and analysis in strategy games (e.g., chess and Go). However, many AI applications are not perceived as AI: "A lot of cutting edge AI has filtered into general applications, often without being called AI because once something becomes useful enough and common enough it's not labeled AI anymore."[2][3]
Various subfields of AI research are centered around particular goals and the use of particular tools. The traditional goals of AI research include learning, reasoning, knowledge representation, planning, natural language processing, perception, and support for robotics.[a] To reach these goals, AI researchers have adapted and integrated a wide range of techniques, including search and mathematical optimization, formal logic, artificial neural networks, and methods based on statistics, operations research, and economics.[b] AI also draws upon psychology, linguistics, philosophy, neuroscience, and other fields.[4] Some companies, such as OpenAI, Google DeepMind and Meta,[5] aim to create artificial general intelligence (AGI)—AI that can complete virtually any cognitive task at least as well as a human.

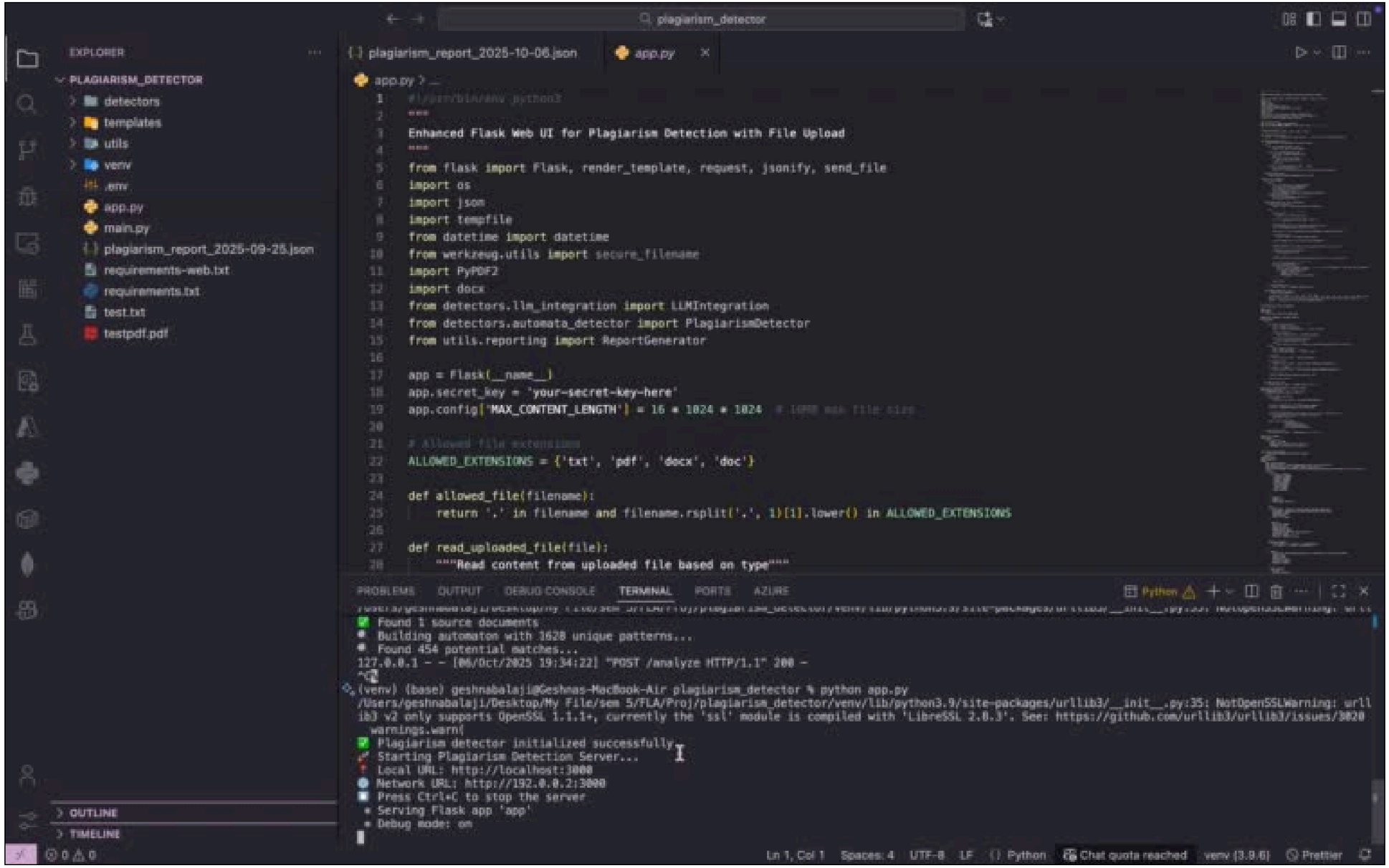
Results


localhost
Artificial intellig...
Input Methods
Text Input
File Upload
Enter text to analyze:
Artificial intelligence (AI) is the capability of computational systems to perform tasks typically associated with human intelligence, such as learning, reasoning, problem-solving, perception, and decision-making. It is a field of research in computer science that develops and studies methods and software that enable machines to perceive their environment and use learning and intelligence to take actions that maximize their chances of achieving defined goals.[1]
Main topic :
artificial intelligence
Helps find more relevant sources for better accuracy
Analyze for Plagiarism
AI-Powered source discovery
Automata theory pattern matching
Wikipedia integration
Analysis Results
99.8%
Similarity
HIGH RISK
Re-analyze
Download Report
Wikipedia Sources Analyzed
Page: Artificial intelligence Summary: Artificial intelligence (AI) is the capability of computational systems to perform tasks typically associated with human intelligence, such as learning, reasoning, problem-solving, perception, and decision-making. It is a field of research in computer science that develops and studies methods and software that enable machines to perceive their environment and use learning and intelligence to take actions that maximize their chances of achieving defined goal...

Results


plagiarism_detector
EXPLORER
PLAGIARISM_DETECTOR
detectors
templates
utils
venv
.env
app.py
main.py
plagiarism_report_2025-09-25.json
requirements-web.txt
requirements.txt
test.txt
testpdf.pdf
plagiarism_report_2025-10-06.json
plagiarism_report_2025-10-06-2.json
test.txt
Extension: vscode-pdf
testpdf.pdf
Automatic Zoom
Hello everyone! hope you are doing well this is our FLA project its plagarism detector
Physics is the scientific study of matter, its fundamental constituents, its motion and behavior through space and time, and the related entities of energy and force.[1] It is one of the most fundamental scientific disciplines.[2][3][4] A scientist who specializes in the field of physics is called a physicist.
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
AZURE
(venv) (base) geshnabalaji@Geshnas-MacBook-Air plagiarism_detector % python app.py
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:3000
* Running on http://192.0.0.2:3000
Press CTRL+C to quit
* Restarting with stat
/Users/geshnabalaji/Desktop/My File/sem 5/FLA/Proj/plagiarism_detector/venv/lib/python3.9/site-packages/urllib3/__init__.py:35: NotOpenSSLWarning: urllib3 v2 only supports OpenSSL 1.1.1+, currently the 'ssl' module is compiled with 'LibreSSL 2.8.3'. See: https://github.com/urllib3/urllib3/issues/3020
warnings.warn(
Plagiarism detector initialized successfully
Starting Plagiarism Detection Server...
Local URL: http://localhost:3000
Network URL: http://192.0.0.2:3000
Press Ctrl+C to stop the server
* Debugger is active!
* Debugger PIN: 119-080-107
OUTLINE
TIMELINE
Chat quota reached

Results


plagiarism_detector
EXPLORER
PLAGIARISM_DETECTOR
detectors
templates
utils
venv
.env
app.py
main.py
plagiarism_report_2025-09-25.json
requirements-web.txt
requirements.txt
test.txt
testpdf.pdf
plagiarism_report_2025-10-06.json
app.py
app.py > ...
"""
Enhanced Flask Web UI for Plagiarism Detection with File Upload
"""
from flask import Flask, render_template, request, jsonify, send_file
import os
import json
import tempfile
from datetime import datetime
from werkzeug.utils import secure_filename
import PyPDF2
import docx
from detectors.llm_integration import LLMIntegration
from detectors.automata_detector import PlagiarismDetector
from utils.reporting import ReportGenerator
app = Flask(__name__)
app.secret_key = 'your-secret-key-here'
app.config['MAX_CONTENT_LENGTH'] = 16 * 1024 * 1024
ALLOWED_EXTENSIONS = {'txt', 'pdf', 'docx', 'doc'}
def allowed_file(filename):
return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS
def read_uploaded_file(file):
"""Read content from uploaded file based on type"""
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
AZURE
Python
Found 1 source documents
Building automaton with 1628 unique patterns...
Found 454 potential matches...
127.0.0.1 - - [06/Oct/2025 19:34:22] "POST /analyze HTTP/1.1" 200 -
(venv) (base) geshnabalaji@Geshnas-MacBook-Air plagiarism_detector % python app.py
/Users/geshnabalaji/Desktop/My File/sem 5/FLA/Proj/plagiarism_detector/venv/lib/python3.9/site-packages/urllib3/__init__.py:35: NotOpenSSLWarning: urllib3 v2 only supports OpenSSL 1.1.1+, currently the 'ssl' module is compiled with 'LibreSSL 2.8.3'. See: https://github.com/urllib3/urllib3/issues/3020
warnings.warn(
Plagiarism detector initialized successfully
Starting Plagiarism Detection Server...
Local URL: http://localhost:3000
Network URL: http://192.0.0.2:3000
Press Ctrl+C to stop the server
* Serving Flask app 'app'
* Debug mode: on
OUTLINE
TIMELINE
Ln 1, Col 1
Spaces: 4
UTF-8
LF
Python
Chat quota reached
venv (3.9.6)
Prettier

Future Work

- Integrate advanced transformer models for deeper semantic detection.
- Use advanced automata (NFA, PDA, Weighted FA) to detect complex or paraphrased plagiarism.
- Combine Finite Automata with AI/LLMs for better semantic understanding.
- Develop adaptive automata that learn and update with new data.
- Implement parallel DFA processing for real-time large-scale detection.
- Create domain-specific automata for topic-aware plagiarism checks.
- Add visual DFA explainability to show why text was flagged.

References

[1] Manning, C., Raghavan, P., & Schütze, H. (2008). Introduction to Information Retrieval. Cambridge University Press. https://doi.org/10.1017/CBO9780511809071

[2] Potthast, M., Barrón-Cedeño, A., Stein, B., & Rosso, P. (2011). Cross-language plagiarism detection. Language Resources and Evaluation, 45(1), 45–62. https://doi.org/10.1007/s10579-010-9132-0

[3] Alzahrani, S. M., Salim, N., & Abraham, A. (2012). Understanding plagiarism linguistic patterns, textual features, and detection methods. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 42(2), 133–149. https://doi.org/10.1109/TSMCC.2011.2161825

[4] Maurer, H., Kappe, F., & Zaka, B. (2006). Plagiarism – A survey. Journal of Universal Computer Science, 12(8), 1050–1084. https://www.jucs.org/jucs_12_8/plagiarism_a_survey

[5] Sharma, S., & Choudhary, A. (2019). Semantic plagiarism detection using Word2Vec and deep learning. International Journal of Advanced Computer Science and Applications, 10(3), 45–52. https://doi.org/10.14569/IJACSA.2019.0100307

Thank you