Keyword Extraction System

(Rule-Based + Machine Learning Hybrid NLP Application)

Submitted By :

M.Kalyani - AP23110010343

B.Jahnavi - AP23110010342

B.Ishwarya - AP23110010646

D.Turvi - AP23110010391

S.L.S.Sanjana - AP23110010295

PROJECT DESCRIPTION

The Keyword Extraction System is an advanced Natural Language Processing (NLP) application designed to automatically extract the most meaningful keywords, concepts, and keyphrases from documents, articles, and investigation reports. The system uses a hybrid approach combining **rule-based methods** and **machine-learning-driven graph algorithms (TextRank)** to analyze text and identify the most important terms. It is built as a user-friendly web application using **Streamlit**, enabling journalists, researchers, analysts, and students to quickly process large volumes of text, generate precise summaries, and visualize key themes using word clouds. By empowering users with automated text intelligence, this system enhances productivity and improves decision-making across multiple domains.

PROJECT SCENARIOS

Scenario 1: Investigative Journalism – Document Analysis

A journalist receives a 30-page confidential report on financial fraud. Instead of manually reading every page, the Keyword Extraction System processes the entire document within seconds. It extracts critical keywords such as "embezzlement," "unauthorized transactions," "shell companies," and generates a summary highlighting the major findings. The journalist instantly gains insights, allowing faster fact-checking, story development, and investigative leads.

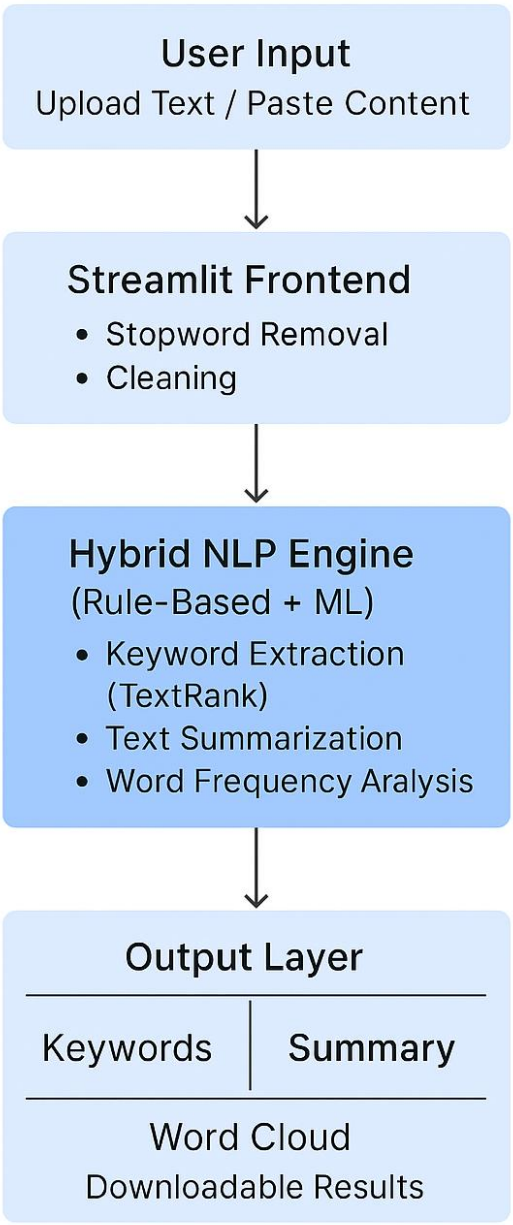
Scenario 2: Research – Literature Review Automation

A postgraduate student is working on a research paper. They upload multiple article excerpts containing dense academic language. The system extracts high-value keywords such as "neural optimization," "model accuracy," "data sparsity," and generates summaries that help the student quickly understand key contributions from each paper. This reduces manual reading time and improves research efficiency.

Scenario 3: Corporate – Business Report Insight Mining

A business analyst receives a 15-page annual report. Using the Keyword Extraction System, they extract organizational priorities, financial terms, trend indicators, and operational objectives. The generated summary highlights growth areas, risk factors, and performance metrics. This enables management to make faster and informed business decisions.

TECHNICAL DIAGRAM



PREREQUISITES

To complete this project, the following software and packages are required:

Software

- Python 3.x
- VS Code / PyCharm / Any IDE
- Streamlit environment

Python Packages

(Installed using pip as in *requirements.txt*

requirements

)

pip install streamlit

pip install summa

pip install wordcloud

pip install matplotlib

PRIOR KNOWLEDGE

To understand and implement this project successfully, the user should have basic knowledge of:

- NLP Concepts
- TextRank Algorithm
- Keyword Extraction Techniques
- Summarization Techniques
- Basics of Streamlit Web Framework
- Python Programming
- Word Clouds & Visualization

PROJECT FLOW

- User uploads a text file or pastes raw text
- System reads and preprocesses the input
- Keywords are extracted using TextRank (Summa)
- Summary is generated
- Word cloud visualization is created
- Outputs are displayed and available for download

PROJECT ACTIVITIES

Milestone 1: Data Input & Preparation

Activity 1.1: Input Acquisition

The system accepts two forms of input:

- Uploaded .txt files
- Manually pasted text inside Streamlit textbox

This allows flexibility and supports large documents.

Activity 1.2: Preprocessing

The system automatically:

- Removes unnecessary characters
- Handles spacing
- Prepares the text for keyword extraction
- Avoids interruption due to formatting issues

Activity 1.3: Validating Input

The application checks:

- File type
- Text length
- Empty or invalid submissions

If text is too short, the function handles it gracefully (as seen in *app.py* extract_keywords() logic)

app

.

Milestone 2: Exploratory Text Analysis (NLP EDA Equivalent)

Activity 2.1: Descriptive Text Overview

Instead of numerical EDA, the system examines:

- Total word count
- Frequency distributions
- Occurrence patterns

Activity 2.2: Tokenization & Stopword Handling

Although Summa handles tokenization internally, the conceptual flow includes:

- Text segmentation
- Graph-building based on word connectivity

Activity 2.3: Word Frequency Distribution

The word cloud represents the most frequent words visually.
Larger words = higher importance.

Activity 2.4: Co-occurrence Graph (TextRank Concept)

The system identifies:

- How words relate
- Their importance rank

- Graph centrality scores

Activity 2.5: Keyword Ranking

Keywords are ranked based on graph-weighted importance.

Milestone 3: Keyword & Summary Generation (Model Building Equivalent)

Activity 3.1: Importing Required Libraries

(As used in your *app.py* file)

app

Activity 3.2: Keyword Extraction Module

Using TextRank:

- Built via Summa library
- Rule-based + graph-based method
- Extracts the top-ranked words

Error handling is included for incorrect input.

Activity 3.3: Summary Generation Module

Also via TextRank summarizer:

- Chooses top sentences
- Maintains coherence
- Produces extractive summary

Activity 3.4: Word Cloud Generation

Using WordCloud + Matplotlib:

- Processes cleaned text
- Generates frequency-based visualization
- Rendered through Streamlit

Milestone 4: Performance Understanding & Method Comparison

Activity 4.1: Rule-Based vs ML Comparison

Rule-Based Methods

- Fast
- Deterministic
- Lightweight

TextRank (Graph ML)

- More accurate
- Captures semantic relationships
- Unsupervised

Activity 4.2: Why TextRank Works Best

- PageRank-inspired
- Word co-occurrence mapping
- Importance assigned by context, not frequency

Milestone 5: Deployment

Activity 5.1: Streamlit Application Architecture

The app is built using:

- Python backend
- Streamlit UI
- Matplotlib rendering
- Download button API
-

BACKEND LOGIC

The backend performs:

- File reading
- Text extraction
- Keyword extraction
- Summary generation
- Word cloud creation

All functions (extract_keywords, generate_summary) are taken from your *app.py* file and executed real-time

app

Backend Implementation (app.py)

Key Features

- Clean UI
- Sidebar input section
- Three-tab layout
    - Keywords
    - Summary
    - Word Cloud

Core Highlights from Your Code

- Error handling
- Modular design
- Interactive buttons
- Download buttons
- Word cloud visualization pipeline

FRONTEND IMPLEMENTATION

UI Components:

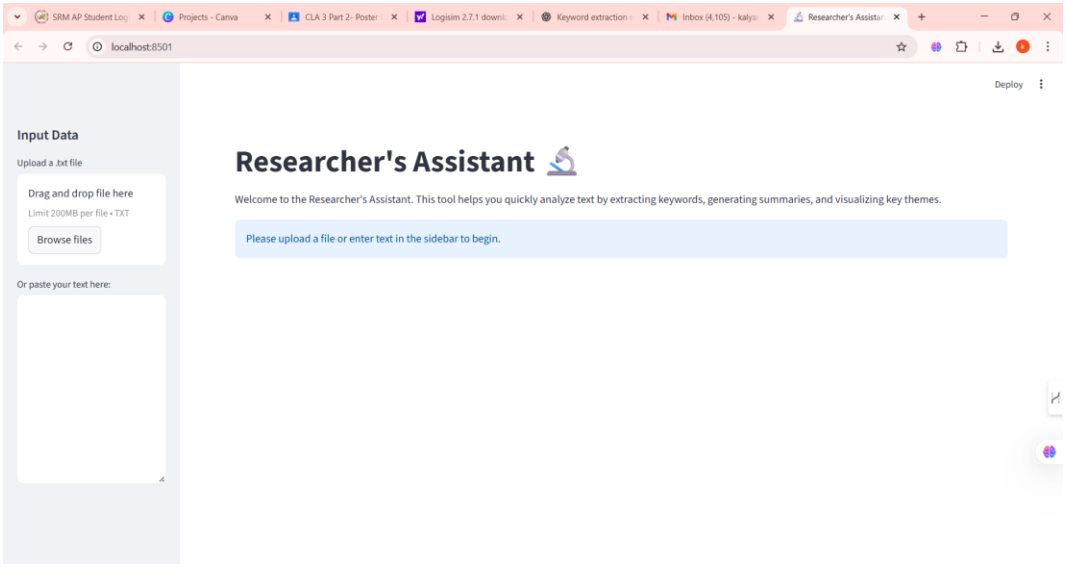
- Sidebar input controls
- Text areas
- Buttons
- Tabs
- Cloud Visualization Canvas

All built using Streamlit.

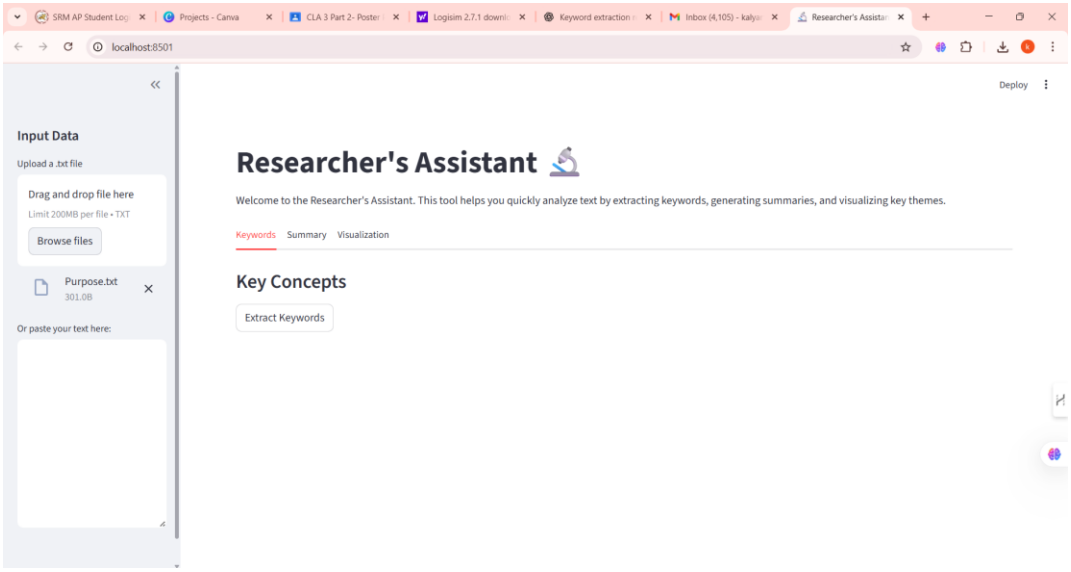
APPLICATION SCREENSHOTS AND WORKFLOW

(Upload your screenshots later, I will replace placeholders.)

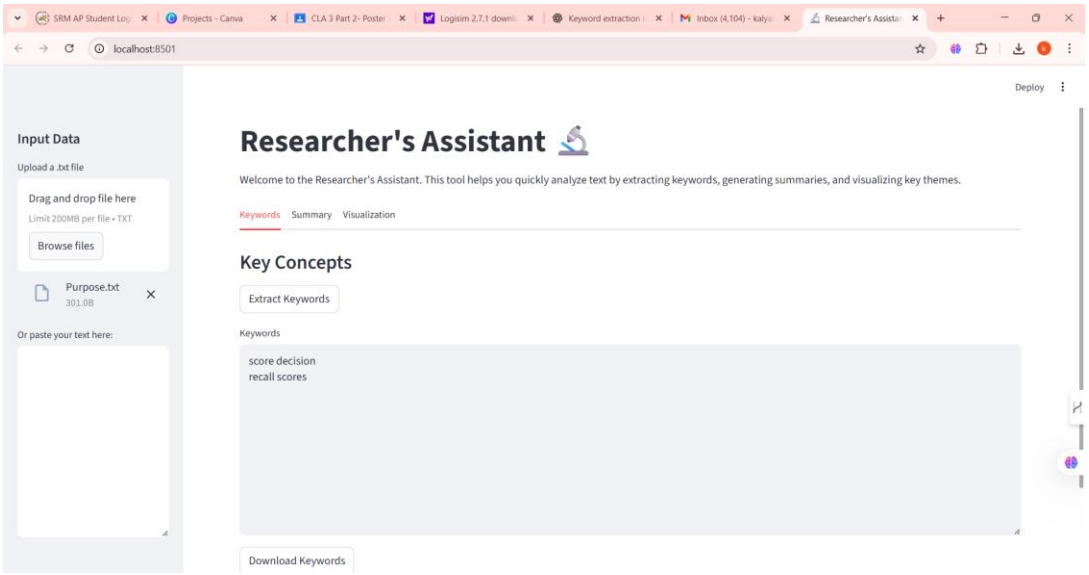
1. **Home Page Interface**



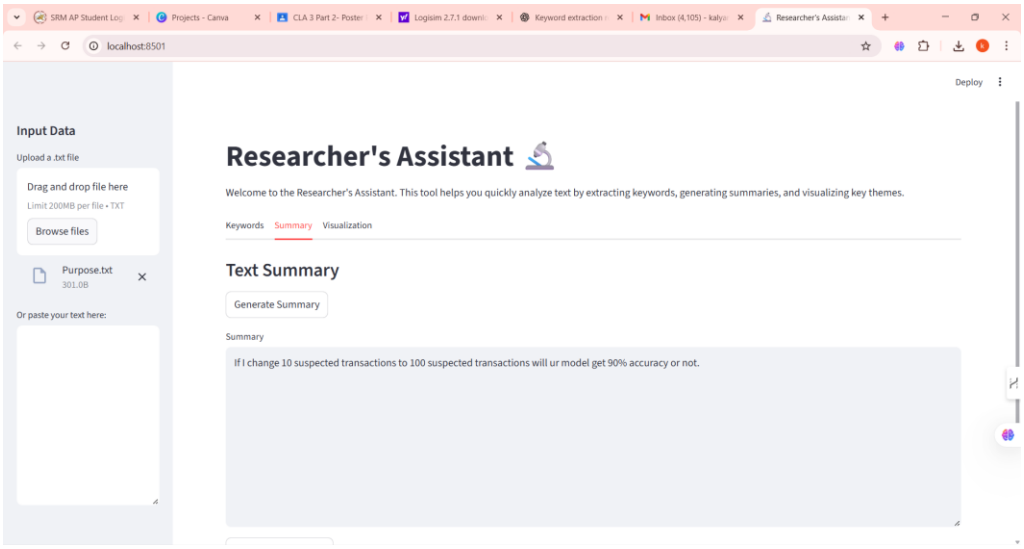
Text Input / File Upload Section



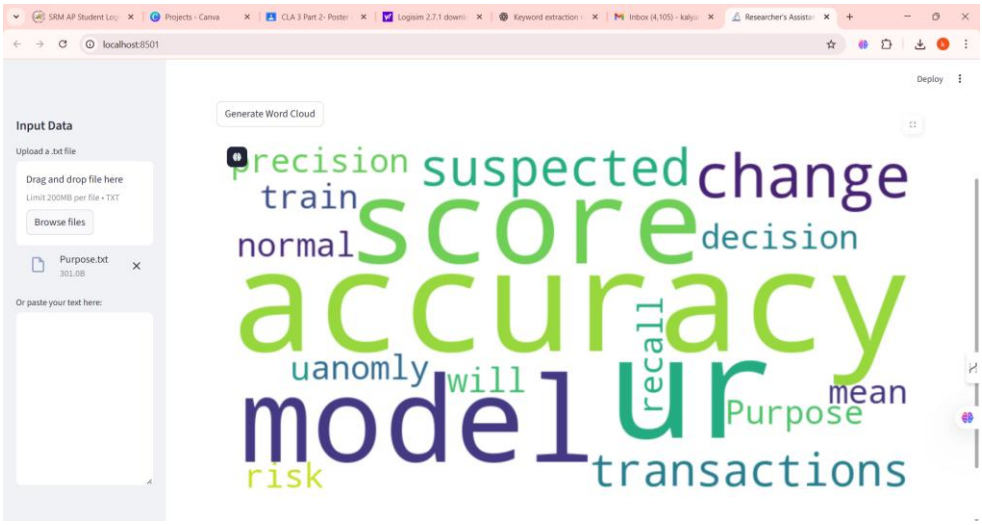
2. **Keyword Extraction Results**



Summary Generation Results



3. **Word Cloud Visualization**



FUTURE IMPLEMENTATIONS

- PDF upload support
- Multi-language keyword extraction
- Support for entire dataset folders
- Export results in multiple formats
- Integrating BERT/Transformer-based models
- Named Entity Recognition (NER)
- Topic Modeling (LDA / BERTopic)
- API deployment with FastAPI

CONCLUSION

The Keyword Extraction System successfully automates the extraction of meaningful keywords, generates concise summaries, and provides visual insights using word clouds. Using a hybrid methodology combining rule-based processing and machine learning (TextRank), the system achieves accurate and efficient text analysis suitable for journalists, researchers, and analysts. The Streamlit-based user interface makes the application accessible, interactive, and practical for real-world use. This project demonstrates the end-to-end integration of NLP techniques into a fully functional web application, completing the journey from concept to deployment.