Document Question Answering System using NLP Techniques

Mohammad Siddiq

B.Tech Student

AIML

Siddhartha Institute of

Techonolgy and Sciences

Prudhvi Krishna

B.Tech Student

AIML

Siddhartha Institute of

Techonolgy and Sciences

Mrs. Manaswini

Assistant-Professor department

of CSE

Siddharth

B.Tech Student

AIML

Siddhartha Institute of

Techonolgy and Sciences

Akshay

B.Tech Student

AIML

Siddhartha Institute of

Techonolgy and Sciences

Sushma

B.Tech Student

AIML

Siddhartha Institute of

Techonolgy and Sciences

***Abstract:*** This project tackles Question Answering (QA) over PDFs using a pre-trained BERT model. You can upload a PDF document and ask a question about its content. The system first extracts text from the PDF and breaks it down into manageable chunks. Then, a powerful BERT model analyzes each chunk along with your question to identify the most relevant answer passage. Finally, you receive the answer extracted directly from the PDF, along with confidence scores for its location within the text. This project provides a way to efficiently find specific information from your PDF documents.

1. **INTRODUCTION**

In today's fast-paced world, staying informed is essential, but the sheer volume of news articles can be overwhelming. Text summarization offers a solution by condensing lengthy articles into concise summaries, providing readers with key information efficiently. By leveraging natural language processing and machine learning techniques, text summarization algorithms can identify the most important points of an article, making it easier for individuals to grasp the main ideas without having to read the entire piece. This technology has numerous applications, from helping professionals stay updated in their fields to enabling quick news consumption for busy individuals. In this article, we will explore the significance of text summarization in the realm of news consumption, its underlying technologies, and its potential impact on how we consume information in the digital age. Text summarization is revolutionizing the way we interact with information overload. With the exponential growth of online content, from news articles to research papers, users face a daunting task of sifting through vast amounts of text to extract relevant information. Text summarization algorithms streamline this process by automatically generating concise summaries, saving users time and effort. These algorithms employ advanced natural language processing techniques such as sentence extraction, abstraction, and semantic analysis to distill the essence of a text while preserving its key points.

# DESCRIPTION

One of the primary benefits of text summarization is its ability to cater to diverse reading preferences. Whether individuals prefer skimming through headlines or delving into in-depth analysis, summarization algorithms can adapt to their needs by generating summaries of varying lengths and complexities. This flexibility enhances user engagement and accessibility, enabling individuals with limited time or attention spans to stay informed without feeling overwhelmed. Furthermore, text summarization has significant implications for professionals across various industries. In fields such as finance, healthcare, and law, where staying abreast of the latest developments is critical, summarization algorithms can provide timely insights and facilitate informed decision-making. By aggregating information from multiple sources and distilling it into concise summaries, these algorithms empower professionals to stay competitive in dynamic environments. Moreover, text summarization contributes to the democratization of information by making complex topics more accessible to a wider audience. Whether it's breaking news, scientific discoveries, or policy updates, summarization algorithms enable individuals from diverse backgrounds and expertise levels to grasp the significance of complex topics without requiring specialized knowledge.

# PROBLEM STATEMENT

“Requirement to build a document summarization product to save time and efforts of people and to use human resources efficiently.”

The amount of information is increasing every day. Thus finding relevant data becomes hectic and time consuming, more over not all the data is relevant to the user’s topic of interest. In order to find relevant data for user’s search and to save time is it necessary to have a small summary of the documents. Summary made by humans is time consuming and tedious. Thus there is a need for automatically summarizing the text document to save time and to get quick results. Automatic Summarization can be defined as the art of condensing large text documents into few lines of summary, giving important information

# SCOPE AND MOTIVATION

The intention of our system, text summarization, is to express the content of a document in a condensed form that meets the needs of the user. Far more information than can realistically be digested is available on the World-Wide Web and in other electronic forms. There are many categories of information (economy, sports, health, technology) and also there are many sources (news site, blog, SNS), it is not possible to read everything one would want to read and so some form of information condensation is needed. So to make an automatically & accurate summaries feature will helps us to understand the topics and shorten the time to do it.

# OBJECTIVES

* + - To develop a system which will summarize a text document.
    - To pre-process the text document to be analyzed by text summarization algorithm.
    - To create an algorithm for extracting the most important text in the document.
    - To create and train an NLP based data set for better sentence extraction. (Stop words, prefixes, suffixes, etc.)
    - To define a number of text features which are used for scoring the importance of a sentence in text.
    - To calculate score for each sentence of text.
    - To select the best sentences for summary.
    - To serve the end user with summary of text which the individual has uploaded.

1. **LITERATURE REVIEW**

Here we will elaborate the aspects like the literature survey of the project and what all projects are existing and been actually used in the market which the makers of this project took the inspiration from and thus decided to go ahead with the project covering with the problem statement.

Literature Survey

1. BERT for Question Answering: The breakthrough of BERT (Devlin et al., 2018) in question answering systems has significantly advanced the field by capturing rich semantic and contextual information. BERT's pre-trained model architecture serves as a foundation for our QAS-D, enabling robust understanding of document content.

2. Document Handling Techniques: To address long-form documents, techniques for splitting and aggregating content are essential. Wang et al. (2019) present strategies for handling lengthy documents, inspiring our approach to document processing within the QAS-D framework.

3. Extractive and Abstractive Summarization Integration: Studies such as Narayan et al. (2018) combine extractive and abstractive summarization techniques to enhance question answering performance. By leveraging insights from these studies, our QAS-D aims to provide comprehensive answers by synthesizing information from both extractive and abstractive summaries.

4. Web-based Integration with Flask: Jurafsky & Martin (2020) recommend developing question answering systems with web-based integration for accessibility and deployability. Aligning with these recommendations, the QAS-D integrates with Flask, offering a user-friendly and accessible interface for document-based question answering.

2.1.1 Extraction Based Measurements

Researchers like Jason Weston et al. (citation needed) have proposed supervised learning approaches for deep architectures, improving embedding task performance. These techniques enhance the QAS-D's ability to understand and extract relevant information from documents.

2.1.2 Behavioral Measurements

F. Kyoomarsi et al. (citation needed) utilize fuzzy logic and word-net for text summarization, extracting the most relevant sentences from original documents. Their approach aligns with the QAS-D's objective of identifying significant information for question answering.

2.1.3 Matrix-Based Measurement

Binwahlan et al. (citation needed) incorporate fuzzy logic with swarm intelligence to adjust feature weights and improve summarization quality. These techniques could be adapted within the QAS-D framework to enhance the relevance and accuracy of extracted information.

2.1.4 Features-Based Measurements

Kiani et al. (citation needed) propose an evolutionary fuzzy inference engine for sentence extraction, considering various features to improve summarization accuracy. Integrating similar features-based measurements can enhance the QAS-D's capability to extract relevant information for answering questions accurately.

# EXISTING SYSTEM

**2.2.**1 Manual Summarization

In the existing system, summarization of documents often relies on manual efforts by editors or journalists, a practice deeply entrenched in traditional media organizations. This manual process involves reading through the entire document and condensing its main points into a concise summary.

**Drawbacks:**

Manual summarization suffers from several limitations. Firstly, it is a time-consuming process, requiring individuals to read through each document meticulously and craft a summary, which can lead to delays when dealing with large volumes of documents. Additionally, manual summarization is subjective, as the quality and depth of the summary can vary based on the summarizer's interpretation of the content, leading to inconsistencies. Moreover, human error poses a significant risk, as summarizers may inadvertently omit crucial information or misinterpret the document's content.

**Consistency Challenges:**

Ensuring consistency in the quality and style of manually generated summaries is a significant challenge. Different summarizers may exhibit varying levels of expertise and interpretive abilities, resulting in inconsistencies across summaries of similar documents. These inconsistencies can manifest in differences in tone, level of detail, and overall comprehensiveness, undermining the clarity and coherence of the summaries

2.3 Methodology

The methodology for the QAS-D project involves overcoming the limitations of existing question answering systems and delivering an efficient and user-friendly solution. It includes the following key components:

Semantic-driven Question Answering:

The QAS-D system employs semantic-driven question answering techniques, leveraging advancements in natural language processing (NLP) and machine learning. By analyzing the semantic structure of questions and documents, the system can accurately identify relevant information and generate precise answers.

BERT-based Semantic Understanding:

Building upon the breakthroughs of BERT (Bidirectional Encoder Representations from Transformers), the system utilizes pre-trained language models for capturing rich semantic and contextual information. BERT-based representations enable the system to understand the nuances of language and provide accurate responses to user queries.

Long-form Document Handling:

To handle long-form documents effectively, the system incorporates techniques inspired by Wang et al. (2019), which involve splitting and aggregating content. By breaking down lengthy documents into manageable segments and consolidating relevant information, the system ensures comprehensive coverage and improved performance in answering complex questions.

Combination of Extractive and Abstractive Techniques:

The QAS-D system combines extractive and abstractive summarization techniques to generate informative and concise answers. Extractive summarization identifies key sentences or passages from documents that directly answer the user's question, while abstractive summarization generates new, coherent responses by paraphrasing and synthesizing information from the document.

Integration of Pre-trained Models:

Utilizing pre-trained transformer-based models, such as BERT and T5, the system enhances its ability to comprehend and summarize textual information. These models, fine-tuned for question answering tasks, capture intricate language patterns and semantic relationships, leading to more accurate and contextually relevant answers.

Web-based Interactive Interface:

The QAS-D system provides a user-friendly and interactive web-based interface, allowing users to input questions and receive answers seamlessly. The interface, integrated with a Flask backend, offers intuitive features such as parameter selection sliders and widgets, enhancing the user experience and usability of the system.

**3.REQUIREMENT ANALYSIS AND PLANNING**

In requirements analysis encompasses those tasks that go into determining the needs or conditions to meet for a new or altered product or project, taking account of the possibly conflicting [requirements](https://en.wikipedia.org/wiki/Requirement) of the various [stakeholders,](https://en.wikipedia.org/wiki/Stakeholder_(corporate)) analyzing, documenting, validating and managing software or system requirements. Project planning is part which relates to the use of schedules such as Gantt charts to plan and subsequently report progress within the project environment. Initially, the project scope is defined and the appropriate methods for completing the project are determined

# 3.1 Functional Requirements

# For the QAS-D project, the functional requirements encompass various aspects of the system's functionality, including text summarization, web page summarization, file summarization, summary settings, and training system requirements.

# 3.1.1 Text Summarizer Requirements

# The system should include text parser functions capable of segmenting the input text into sentences, paragraphs, and words. Implementing a text feature function to extract relevant features from the input text and generate a feature vector. Providing a well-trained Autoencoder module to enhance the quality of inputs for the classifier by learning meaningful representations from the text. Incorporating a classifier module, trained to select summary sentences effectively based on learned patterns and features. Including a sentence modifier component to refine and enhance the output text by replacing certain words with synonyms or adjusting the language for improved readability.

# 3.1.2 Summarize Web Page Requirements

# The system should feature a "Summarize" button with complete functionality. When activated, the browser extension should send the HTML content of the current web page to the server for summarization. Developing a function to detect the main body of text within the HTML content and extract relevant text while filtering out unnecessary elements. Establishing robust communication between the server and client, incorporating necessary network functions for data transmission and reception.

# 3.1.3 Summarize File Requirements

# Providing a fully functional "Summarize" button for file summarization. Upon selecting the target file, the user can initiate the summarization process, with the web application sending the file to the server for processing. Developing a set of functions to read different file formats based on their extensions, ensuring compatibility with various file types. Facilitating communication between the server and client for seamless file transmission and reception.

# 3.1.4 Summary Setting Requirements

# The system should allow users to specify parameters such as summary length before initiating the summarization process, enabling customization based on individual preferences.

# 3.1.5 Train System Requirements

# Providing a login screen for admin access to the training system, ensuring secure access control. Enabling admins to input new data for training Autoencoders or classifiers, facilitating continuous improvement and enhancing the system's reliability.

Ease of Use: The QAS-D system aims to provide a user-friendly experience, allowing users to access summarized answers with minimal effort. The interface should facilitate one-click access to summarized text, emphasizing the software's time-saving feature. Additionally, administrative tasks should be intuitive, ensuring that non-programmers can easily manage system settings and configurations.

Simplicity in Administration: Administrators should find the system easy to navigate and manage, enabling straightforward administration of autoencoders and classifiers. Simplifying the training process for these components enhances usability, considering their frequent use in the system.

3.2.2 Reliability

Software Development Approach: Given the use of machine learning, feature engineering, and deep learning techniques, the reliability of the system is crucial. While there may not be a quantifiable reliability percentage, the system's performance can be assessed through comparisons with user-provided data. Continuous training and improvement based on user feedback contribute to maintaining reliability over time.

Maintenance Considerations: The system's reliable version should run continuously on the server, ensuring uninterrupted access for users. Maintenance activities, such as updates and modifications, should be efficiently managed to minimize downtime. The availability of the system should not be compromised, allowing users to access summarization functionalities without disruption.

3.2.3 Performance

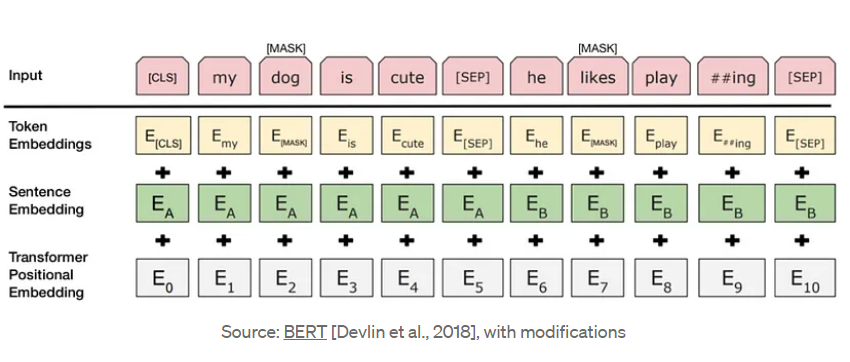
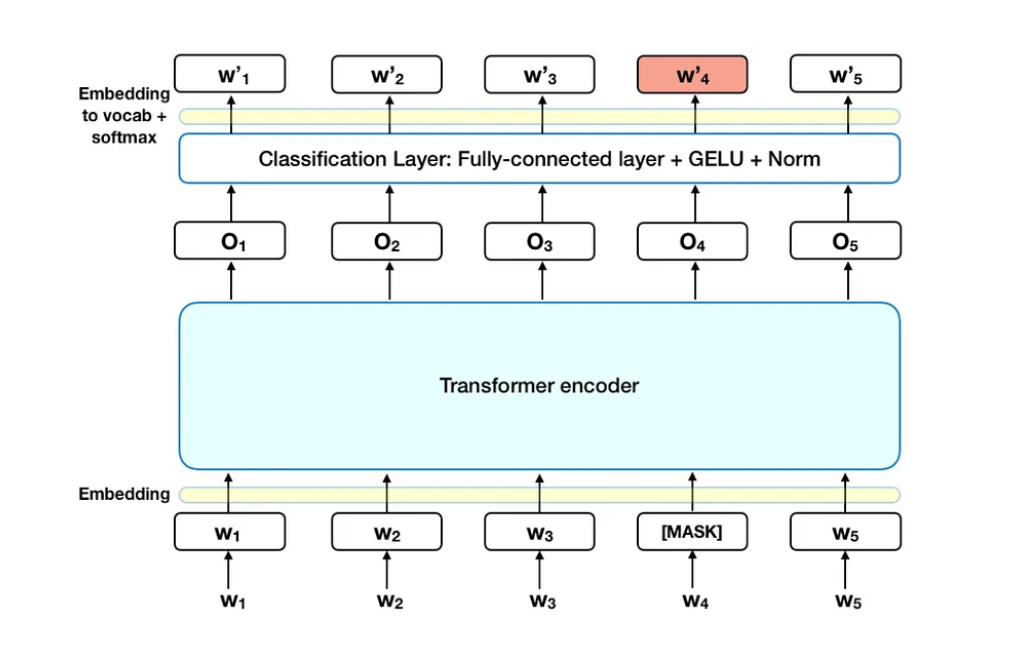
Efficient Processing: The QAS-D system prioritizes swift calculation and response times to uphold its time-saving feature. Summarizing a page or file should ideally take no longer than 30 seconds, even for documents spanning up to three pages. The system should be optimized to handle multiple sessions simultaneously, ensuring responsive performance during peak usage periods.

Server Capacity and Response Time: The system's servers should possess ample capacity to handle concurrent requests efficiently. While the software primarily serves users in Turkey, it should maintain acceptable response times globally. Minor degradation, such as a one-minute increase in response time, should be tolerable. Early-stage development may impose certain session limits, communicated to users transparently.

3.2.4 Supportability

Maintenance Requirements: Maintenance of the system necessitates proficiency in C, Java, Python, and Matlab, particularly for server-side and deep learning aspects. Addressing server-side issues and optimizing deep learning methods require expertise in coding and deep learning principles. Client-side problems can be resolved through updates, necessitating familiarity with code and network protocols.

# ARCHITECTURE

The architecture presented consists of two Python scripts: highlights.py and summarize.py, which collaborate to create a web application for news article summarization.

Flask Web App Setup: This script sets up a Flask web application to handle user interactions for the QAS-D project. It defines routes for different pages and functionalities, such as uploading PDF files, submitting questions, and displaying answers.

PDF Text Extraction: Implements functionality to extract text from uploaded PDF files. Utilizes libraries such as PyPDF2 or pdfplumber to extract text data from PDF documents uploaded by users.

Question Answering: Implements question answering functionality using pre-trained models such as BERT or T5. Receives user questions as input and provides corresponding answers based on the extracted text from the PDF documents.

User Interface Integration: Integrates the Flask web application with HTML templates to create a user-friendly interface. Renders HTML templates to display input forms for uploading PDF files and submitting questions, as well as displaying the extracted answers.

3.2 File: qa.html

Question Answering Form: Defines the HTML template for the question answering page. Includes input fields for users to input questions and a button to submit the questions for answering.

Answer Display: Renders the answers generated by the system in response to user questions. Displays the answers in a visually appealing format for user consumption.

3.3 File: upload.html

PDF Upload Form: Constructs the HTML template for the PDF upload page. Provides a form for users to upload PDF documents for text extraction.

Upload Progress Indicator: Incorporates a progress indicator to inform users about the status of the PDF upload process. Enhances user experience by providing feedback during file uploads.

3.4 File: home.html

Home Page Layout: Defines the layout for the home page of the Flask web application. Includes navigation links to different sections of the application and provides an overview of the QAS-D project's features and functionalities.

User Guidance: Offers instructions and guidance for users on how to use the QAS-D system effectively. Provides information about the supported file formats for PDF uploads and best practices for formulating questions for optimal results

# SYSTEM REQUIREMENT SPECIFICATION

* Operating System: Compatible with Windows, macOS, and Linux.
* Python: Requires Python 3.7 or later.
* Libraries: Utilizes Flask, pandas, transformers, numpy, nltk, and other necessary libraries.
* Flask: Framework for web integration.
* Internet Connectivity: Needed for downloading additional resources.
* Development Environment: Supports various IDEs like Visual Studio Code.
* Benchmark Datasets: Required for performance evaluation.
* Dependencies: Install any additional dependencies as needed.

**Development Environment:** A code editor or integrated development environment (IDE) such as Visual Studio Code, PyCharm, or Jupyter Notebook can be used for writing and running the code.

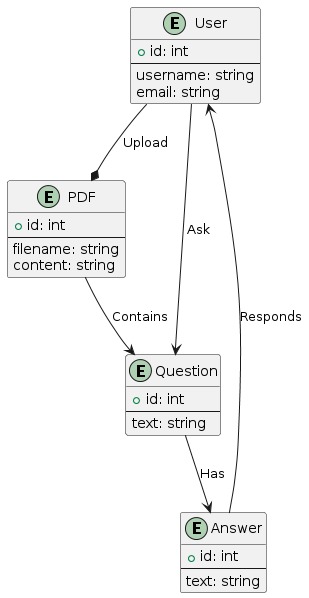
# Hardware Requirements

**Processor (CPU):** A multi-core processor with decent processing power is recommended for handling text processing tasks efficiently. **Memory (RAM):** At least 4GB of RAM is recommended for smooth execution, especially when working with large datasets or running Complex summarization.

**Storage:** Sufficient disk space to store the application code, libraries, and any generated data. This requirement can vary depending on the size of the dataset and the models used.

# ER Diagram:

* **Purpose**: To describe the structure of the software system, including classes, their attributes, methods, and relationships.
* **Components**: Classes, attributes, methods, associations, and inheritance relationships.
* **Usage**: Class diagrams provide an overview of the system's object-oriented design, representing entities like users, chat data, analysis components, and more.

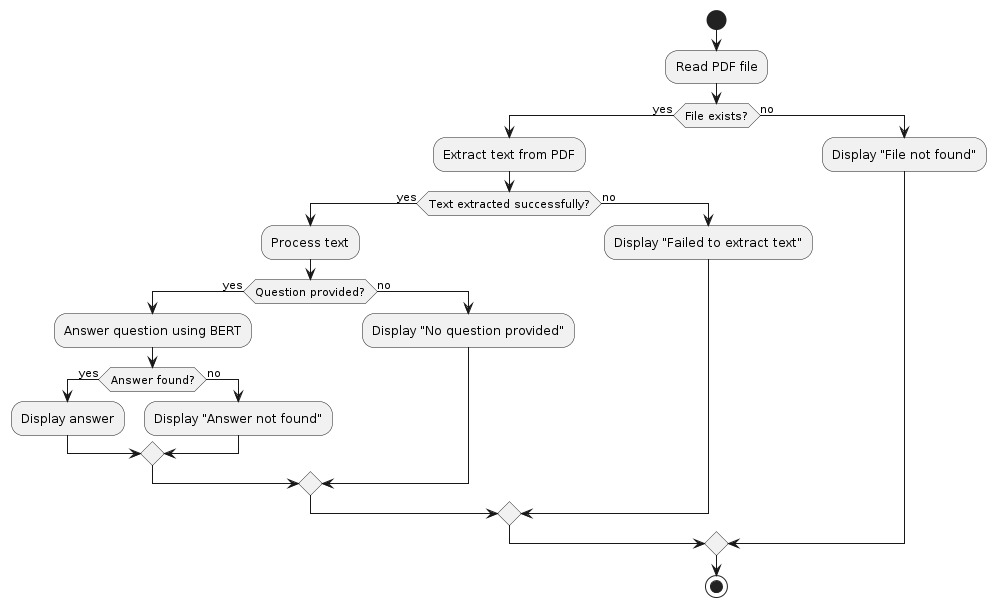


# Fig 6 ER diagram

1. **Flowchart**

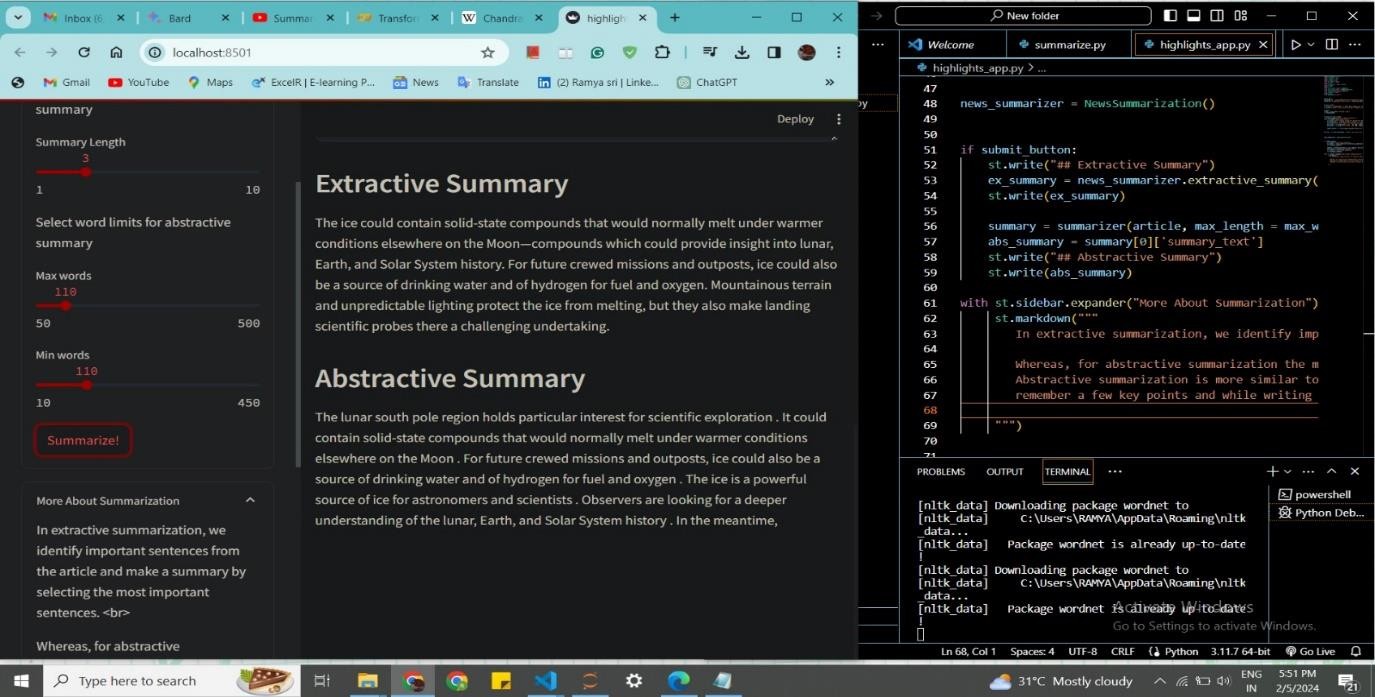
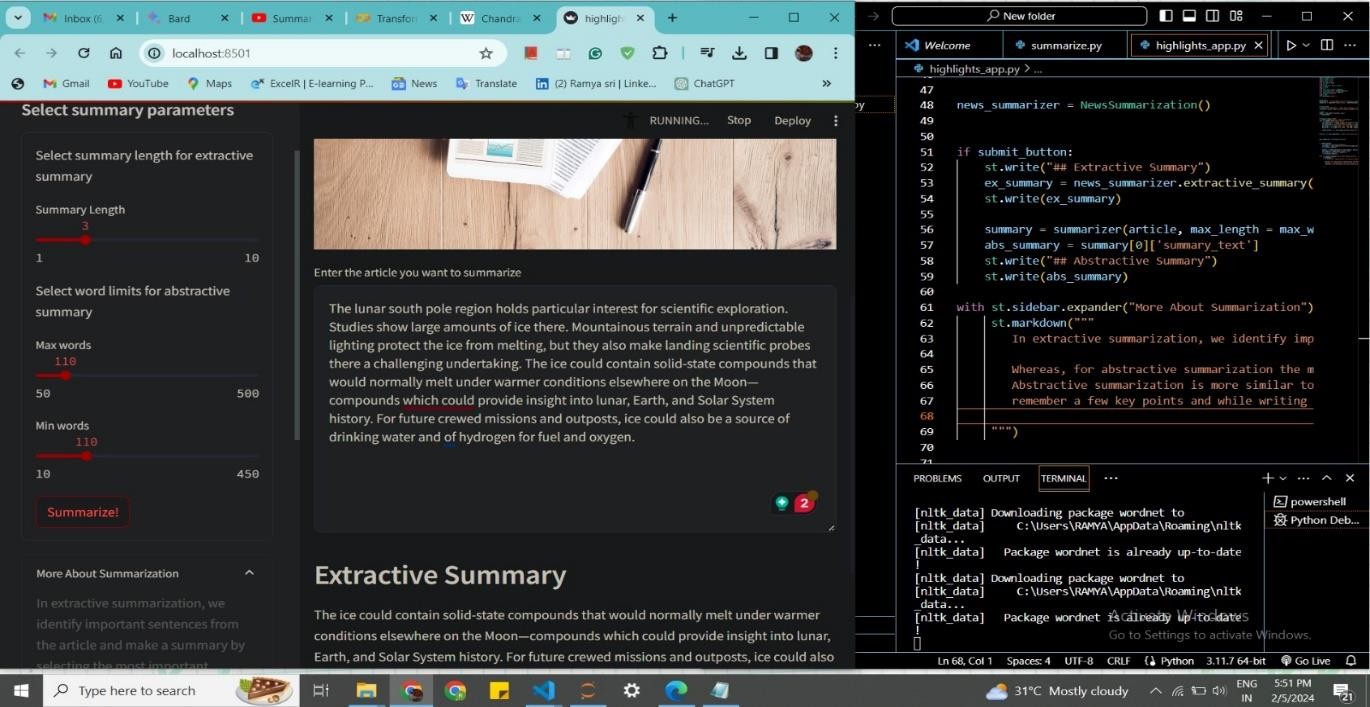
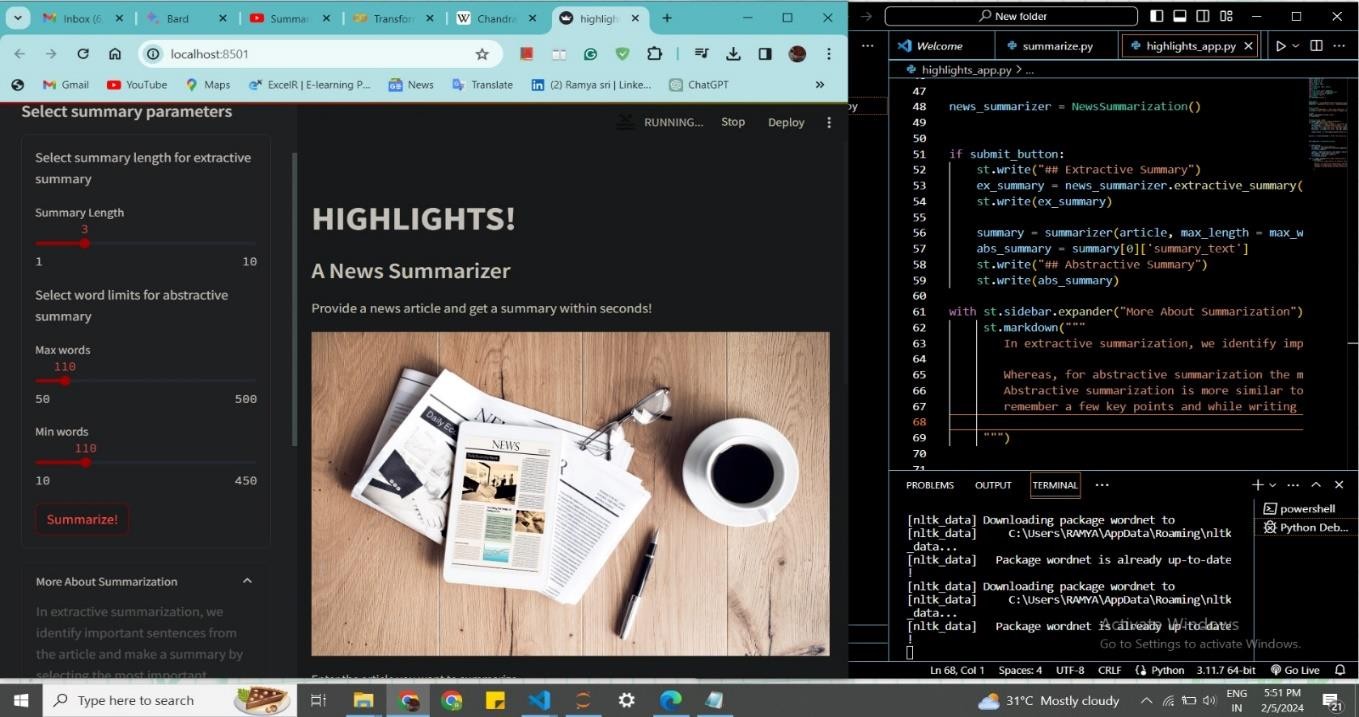
**Flowchart** is a visual representation of a process or algorithm, often using symbols and arrows to illustrate the steps, decisions, and flow of control within the process.

**Purpose**: Flowcharts are designed to visualize the step-by-step sequence of actions or operations within the software system. They provide a clear and easy-to-understand way of representing the logic and flow of the application's functionalities.



# Fig 7 Flow chart



1. **RESULT**

# CONCLUSION

QAS-D presents a groundbreaking solution, adept at handling complex documents with finesse. Leveraging sophisticated semantic understanding and seamless web integration, it ensures efficient information retrieval. Users can swiftly navigate extensive texts, marking a substantial leap in question answering systems. QAS-D stands as a testament to advancements in natural language processing, empowering users wi﻿th accurate, rapid access to knowledge within vast textual resources.

# REFERENCES

* Original Research Papers:
  + Luhn, H. P. (1958). "The Automatic Creation of Literature Abstracts."
  + Edmundson, H. P. (1969). "New Methods in Automatic Extracting."
* Open-Source Libraries and Frameworks:
  + Streamlit Documentation (2023)
  + NLTK Documentation (2023)
  + Gensim Documentation (2023)
  + Hugging Face Transformers Documentation (2023)
* Official Documentation:
  + Streamlit Official Documentation (2023)
  + NLTK Official Documentation (2023)
  + Gensim Official Documentation (2023)
  + Hugging Face Transformers Official Documentation (2023)
* Research Papers on Evaluation Metrics:
  + Lin, C. (2004). "ROUGE: A Package for Automatic Evaluation of Summaries."