A logo with a number and text

Description automatically generated with medium confidence

Final Report on

AI-Powered Document Summarizer

By

Anvit Gupta

22114009

9462011044

[anvit\_g@cs.iitr.ac.in](mailto:anvit_g@cs.iitr.ac.in)

Krishna Aggarwal

22114048

7814355765

[krishna\_a@cs.iitr.ac.in](mailto:krishna_a@cs.iitr.ac.in)

Sarvasva Gupta

22114086

7989904811

[sarvasva\_g@cs.iitr.ac.in](mailto:sarvasva_g@cs.iitr.ac.in)

Utkarsh Lohiya

22114103

7905692105

[utkarsh\_l@cs.iitr.ac.in](mailto:utkarsh_l@cs.iitr.ac.in)

Vineet Kumar

22114107

9263436403

[vineet\_k@cs.iitr.ac.in](mailto:vineet_k@cs.iitr.ac.in)

Contribution

The respective contributions of various team members are as follows:

1. Anvit Gupta

* Project Idea of AI Powered Document Summarizer (Summarix)
* Deciding the skeleton of the application
* Complete design document of the application (with use case and flowchart diagram)
* Website framework using python flask:
  + Extracting text from the input text or document
  + Database integration
  + Email support in contact form
  + Support for size factor, mode, and type of summary
  + Different types of document inputs
* Extractive summary python code without compromising for size, mode and type of summary.
* Try for deployment and hosting.
* Final presentation (40%) of the project
* Complete the readme file for the project.
* Add contributions of the all the members in the final report.

1. Krishna Aggarwal

* Best model selection for abstractive summary generation.
* Abstractive Summarization python code with support for:
  + Paragraph and key points type summary.
  + Different size summary
* Modified extractive python code to include support for:
  + Preprocessing
  + Key points and paragraph summary
  + Different size summary
* Backend final report

1. Vineet Kumar

* Deciding the skeleton of the application
* Responsive design of About page
* Animations to the final presentation

1. Utkarsh Lohiya

* Deciding the skeleton of the application
* Home page (partially executed responsiveness)
* Initial footer design
* Feature section
* Made this Final Report
* Made the Package Diagram

1. Sarvasva Gupta

* SRS document for the project.
* Made initial UI design using skeleton and completed wireframing and prototyping on Figma and finalized the design.
* Decided the logo, color theme, fonts, background images, text to be written on the various portions of the website.
* Implemented JavaScript (front-end) files for ALL the pages.
* Implemented the following using CSS and JS only-
  + Navigation bar – fully responsive and fixed on top from scratch.
  + Carousel display style for team members from scratch.
  + Custom radio buttons which cannot be made directly using CSS properties along with the pop-up from scratch.
  + Scroll-animations in the home page i.e., animation appears on reaching 20% of the element and Reload Animations.
* Tested and Fixed ALL the bugs related to front-end.
* Make CSS files modular and reusable (quality improving) (example: section1.css and section2.css files in github repository).
* Implemented the input and output pages – fully responsive from scratch.
* Final presentation (40%) of the project.

Problem Specification

The "AI Document Summarizer" project is born out of the recognition of a significant problem in today's information-rich world. The problem at hand is the ever-increasing volume of text-based content that individuals and organizations need to navigate and extract valuable insights from. This content encompasses a wide range of sources, including research papers, news articles, legal documents, business reports, and more. The sheer volume of textual information poses a substantial challenge when it comes to quickly comprehending and extracting key insights from lengthy documents. The project's problem specification can be broken down into the following key points:

**1. Information Overload:** In the digital age, access t o information is more abundant than ever before. However, the sheer volume of content can be overwhelming, making it difficult for individuals and organizations to keep up with, understand, and make informed decisions based on the wealth of information available.

**2. Time-Consuming Document Review:** Traditional methods of reviewing lengthy documents are time-consuming and labor-intensive. Manually reading and summarizing documents can be a painstaking process that diverts resources away from other essential tasks.

**3. Lack of Automated Solutions:** While there are manual summarization techniques and tools available, the need for an automated, AI-powered document summarizer remains unmet. Automated solutions can significantly improve efficiency, productivity, and the ability to quickly extract valuable insights from documents.

**4. Varied Document Types:** The problem extends to diverse document types, from research papers loaded with technical jargon to legal documents and business reports filled with complex language. A universal solution capable of handling this variety is required.

**5. Maintaining Quality:** Automated summarization should not come at the expense of quality. The challenge is to develop a system that can automatically generate summaries that are not only concise but also coherent, reliable, and contextually accurate.

**6. Customization:** Recognizing that different users have unique needs when it comes to document summarization, the project aims to provide customization options, such as specifying the desired length and depth of the summary.

**7. Accessibility and Ease of Use:** The project aims to develop a user-friendly interface that can be easily accessed and operated by a wide range of users, from individuals seeking to save time on document review to professionals and organizations looking to streamline information management.

**The problem specification is not limited to the challenges faced by a particular industry or domain but is broad-reaching, touching on the needs of academia, research, journalism, legal practice, business, and beyond. The "AI Document Summarizer" project aims to address these challenges by developing a powerful tool that leverages AI and NLP technologies to efficiently and effectively extract crucial knowledge from a sea of text-based content. This solution is expected to have a substantial impact on improving information management and decision-making processes for a wide range of users and applications.**

Overall Requirements of the Project

The successful execution of the "AI Document Summarizer" project requires a comprehensive set of technical and functional requirements. These requirements are essential to ensure that the project delivers a robust, user-friendly, and efficient document summarization tool. Here is an overview of the overall requirements of the project:

**1. User Interface (UI):**

* Intuitive Design: The user interface should be intuitive, easy to navigate, and visually appealing to accommodate users with varying levels of technical expertise.
* Responsive Design: The system's user interface must be responsive, ensuring that it functions smoothly and adapts to various screen sizes and devices.

**2. Document Input:**

* Support for Multiple Formats: The system should accept documents in multiple formats, including PDF, DOC, DOCX, and plain text.
* Paste Text Option: Users should be able to paste text directly into the system for summarization, in addition to file uploads.

**3. Customization:**

* Summary Length Options: Users should have the ability to customize the length and depth of the summary, choosing from options like short, medium, or long summaries.

**4. Summarization Algorithms:**

* NLP Integration: The project must implement Natural Language Processing (NLP) algorithms for document summarization, ensuring that the system can understand and extract key information from the input documents.
* Efficiency: Summarization algorithms should be optimized for performance to generate summaries within a reasonable timeframe.

**5. Readability Enhancement:**

* Coherent Summaries: The system should ensure that the generated summaries maintain readability by organizing content logically and coherently.
* Context Preservation: Summaries should maintain the original context and flow of the document to facilitate an accurate understanding.

**6. Error Handling:**

* Robust Error Handling: The system should have effective error handling mechanisms in place to gracefully manage scenarios where users input invalid documents, unsupported formats, or encounter processing issues.
* User-Friendly Error Messages: Users should receive informative and user-friendly error messages that guide them in troubleshooting issues.

**7. User Feedback System:**

* Feedback Mechanism: The system should provide a feedback mechanism, allowing users to comment on the generated summaries and report any issues.

**8. Documentation:**

* Comprehensive Documentation: The project should be accompanied by clear and comprehensive documentation that guides users on effectively utilizing the tool and provides insights into its functionality.

**9. Performance and Speed:**

* Timely Summarization: Summarization results should be generated within a reasonable timeframe, even for large documents, to ensure that users do not experience significant delays.

**10. Technical Stack:**

* Technology Selection: The project will leverage Python for backend development, NLP libraries such as NLTK or spaCy, Flask for the server setup, React-JS for the front end, and CSS for styling.

**These overall requirements collectively define the core functionalities and performance expectations of the "AI Document Summarizer" project. Meeting these requirements is essential for delivering a reliable, efficient, and user-friendly document summarization tool that addresses the challenges posed by information overload in today's information-rich world.**

Highlights

The "AI Document Summarizer" project is a remarkable endeavor that has achieved several notable highlights, showcasing its innovation and significance in the field of document summarization. These highlights encompass key achievements, features, and contributions that make the project stand out:

**1. Customization and User Flexibility:**

The project provides users with the unique ability to customize the length and depth of summaries, catering to their specific requirements. This customization feature enhances user flexibility and utility.

**2. Generation of both Extractive and Abstractive Summaries:**

The project is well suited to generate both types of summaries – be it Extractive or Abstractive. This is ensured to provide user’s the choice to switch between quicker summaries and well researched summaries at their will.

**3. Readability and Context Preservation:**

The project's focus on readability and context preservation is a standout feature. It ensures that the generated summaries are not only concise but also coherent and aligned with the original document's flow.

**4. Effective Error Handling and Feedback Mechanism:**

The project's robust error handling mechanisms and user-friendly feedback system contribute to a seamless user experience. Users can confidently use the tool and report issues or provide feedback for continuous improvement.

**5. Able to produce the summaries in both paragraph and bullet points format:**

The project's output can be switched to generate summaries either in a series of paragraphs or in the form of a collection of bullet points. This feature enhances the user spectrum of our project as it caters to the needs of both types of users.

**5. Transparency through Word Counts:**

The project's transparency feature, displaying word counts for both input documents and generated summaries, empowers users with insights into the summarization process and promotes transparency.

**6. User-Centric Design:**

The user-centric design of the project ensures an intuitive and responsive user interface, making it accessible to users with varying levels of technical expertise.

**These highlights collectively demonstrate the project's innovation, user-centric approach, and potential to address the challenges of information overload. The "AI Document Summarizer" project offers an efficient, reliable, and accessible solution for individuals and organizations seeking to manage and extract crucial insights from vast amounts of text-based content.**

Technical Know-How

The output of the "AI Text Summarizer" project is distributed between two fields: whether to generate abstractive summary or extractive summary.

**A) For Extractive Summary:**

1. Objectives:

The primary objective was to calculate the frequency of words in the given text. Then use this frequency table for scoring the sentences in the text. If the **score of the sentence** is greater than a **threshold value** only then it will be included in the summary.

1. Libraries and Components:

It uses the NLTK library from which we use the following components:

1. **Punkt:** It is a module that is used in tokenization i.e. to tokenize the text into words and sentences.
2. **stopwords:** It is used to remove part of speech.
3. **re:** for removing punctuation and extra spaces in the text.
4. Data Preprocessing:

Raw data undergoes extensive preprocessing, including text cleanings of various grammatical constructs like the parts of speech, punctuation, and formatting.

1. Summary Processing:

The summary can be generated in the form of:

1. Paragraph
2. Bullet Points

**Size**: The size of the summary can be varied by varying the threshold value which determines which sentences are to be taken. For high values of threshold, the summary would be short as it will be constructed by picking only those sentences that have high scores. Similarly, for low values of threshold, the summary would be large as will pick from a greater pool of sentences.

**B) For Abstractive Summary:**

1. Objectives:

The primary objectives were as follows:

* Using a pretrained transformer model
* Give a good quality summary in minimum time

1. Model Development:

The AI Text Summarizer model was designed using a **transformer-based architecture.** **(google/flan-t5-base)**

Functions Used:

* **Pipeline function** which is present in **Transformer** library is used to create a NLP pipeline by feeding the **transformer model** and the **specified task i.e. Summarization.**
* We are required to pass the input properly and then choose all the **nlp activities** like **preprocessing and passing the preprocessed** to the transformer and then further generation of the summary will be taken care of by our **pipeline object.**
* Summary can be generated in the form of:

1. Paragraph
2. Bullet Points

**Size:** The size of the summary can be varied by varying the length of the **epoch** and by selecting the **max length of the summary that can generated for a single epoch**.

**Challenges Faced:**

Challenges during the project included summary quality, model complexity, etc. For the model with larger complexity, the time required for summary generation is huge but the quality of the summary generated is higher. So both factors have to be optimized simultaneously.

Backend Design Specifications of the Project

The Backend Design Specifications section will take you through a tour of the internal workings of our project just like how the Narration Video attached described the superficial looks. So, let’s begin:

**THE MAKING OF EXTRACTIVE SUMMARY.PY**

1. First of all let us have a look at the **dependencies** imported to make our project idea into a reality:

**A black background with white text

Description automatically generated**

**A black background with text

Description automatically generated**

2. Then we have an error handling import:

**A black screen with white text

Description automatically generated**

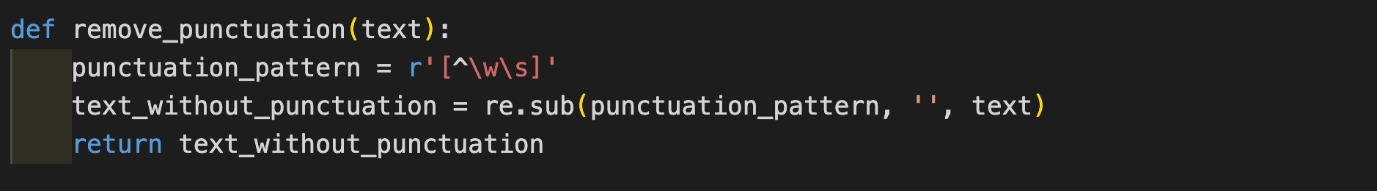
3. Then we have included a function “remove\_punctuation” is created to remove punctuation in the text:

Firstly, a variable “punctuation\_pattern” is defined to store all the punctuations.

Then a variable “text\_without\_punctuation” removes all the characters in the input text

which are present in the “punctuation\_pattern”.

Finally, the function returns “text\_without\_punctuation”.

****

4. We have also created a “summarize” function for summarization of the previously processed input.Using this function , firstly, we remove the punctuations and create a variable named “stopwords” for storing all the stopwords of English. And by using “word\_tokenize” we will split the text into a bunch of words and store them in the variable titled “words” shown below:

**A screen shot of a computer code

Description automatically generated**

5.We have created another variable named “k” for setting up the “threshold value” for different sizes. This helps in producing summaries of varied lengths.

**A screenshot of a computer screen

Description automatically generated**

6. Next, we have created a “freqTable” map with all the words of the dictionary which are initially automatically mapped to zero. Now we iterate over the words in the input text. If a word is not a stopword then it’s frequency in freqTable would be incremented else nothing will happen. This helps in prominently selecting the desired words of considerable significance while neglecting the unnecessary ones at the same time:

**A screen shot of a computer program

Description automatically generated**

7.Next step is to iterate over the sentences. This calculates the scores of the sentences by adding the frequencies of individual words present in that sentence after excluding the stopwords. After this, the average score is reported:

**A screen shot of a computer program

Description automatically generated**

8.Lastly, the variable declared with the name “summary” is used to store the extractive summary. A variable “ct” is also used side-by-side for setting the length of the paragraphs.

For this final step, we are basically iterating over all the sentences. And if the score of the sentence is greater than a threshold value i.e. k\*average then this sentence is appended in the summary, otherwise rejected. But before appending extra spaces and new lines are removed .

If style = = “keyPoints” , then the sentence is appended in the next line in summary with a bullet point.

Else if style = = “paragraph” , then sentence would be added to the summary, During this step, ct is incremented if ct ≠ 5 but if ct = = 5 , a new line is introduced instead and ct is set to zero.

**A computer screen shot of a computer code

Description automatically generated**

**THE MAKING OF ABSTRACTIVE SUMMARY.PY**

1.Firstly, we have created a “summarize” function for generating an abstractive summary which has parameters “input” i.e. our input text, “size” i.e. size of summary required by the user, “pipe” which is an object of nlp pipeline which is set with summarization task and our transformer model = “google/flan-t5-base”, “style” i.e. format of summarization keypoints or paragraph. Now a “step” variable is created for setting the length of the “epoch”. The “max\_summary\_length” variable is used to set the max length of the summary that can be generated when an epoch of passed into “pipe”. For different sizes, different sets of values are created for variables “step” and “max\_summary\_length” to generate different length values of summary for the same input.

Here is the implementation:

**A screenshot of a computer program

Description automatically generated**

2. Next**,** we have declared an “output” variable for storing abstractive summary. Now we are iterating in input. we have created a var “seg” in which our epoch i.e. part of input which is passed in pipe in one iteration. variable “k” and “l” are initial and final indices of epoch and then they are increment by a value of “epoch” or length of epoch for giving the next epoch in the next iteration. And then summary is generated for that epoch by passing it in the pipe object. The variable “min\_length” sets the minimum length for the summary generated. Now this summary is appended to the output. Finally, the variable “sum\_length” stores the length of the total summary of the whole input. Here is the talked implementation:

**A computer screen shot of a program code

Description automatically generated**

If style == “keyPoints” , then output is splitted across the character ‘.’ and then lines are joined by adding a new line and bullet point character. The last character of the last line is omitted since it is a bullet point which is not required. And a bullet point is appended at the start.

Else if style == “paragraph” , a variable “c” is created for setting the length of the paragraph. And then a list is created of sentences present in the “output” variable and then we iterate in the list then sentence in list would be added to summary and c is incremented if c ≠ 5 but if

c == 5 then a new line is introduced and c is set to zero. After all this, the output is returned as shown below

:

**A screen shot of a computer screen

Description automatically generated**

**THE MAKING OF MODEL.PY**

This file contains the model for abstractive summarization. For this, we have imported the pipeline from the transformers.

“model\_ckpt” contains our transformer model “google/flan-t5-base”.

We have also created an NLP pipeline object “pipe” by feeding the transformer model and specified task i.e. Summarization:

**A screen shot of a computer

Description automatically generated**

Initialization Client Information

A computer screen shot of a program code

Description automatically generated

Whenever ‘/’ end point is accessed, methods allowed are GET/POST. For GET request data is sent back to the used in form of index.html whereas POST data is sent to the server and info is stored in database.

A screen shot of a computer screen

Description automatically generated

Whenever the /about end point is accessed, about.html page is rendered.

A computer code with colorful text

Description automatically generated

This section of code is used for running the flask application on localhost:8000

A computer screen shot of text

Description automatically generatedWhenever /input page is accessed, posting requests (hitting summarize button) results in summary generation as per the options/modes/size chosen by the user, finally input.html or output.html is rendered to the user.

Design and Organization of the CSS Files

* **utility.css file**: This file contains colors, font families, font-sizes as variables and, animations, scroll-bar design, pop-ups, which are used throughout the website. This file has been imported into all pages to make the text appearance consistent.

A computer screen shot of text

Description automatically generated

For example, look at the above code; the variables declared in utility.css have been directly used, which offers flexibility to make large changes to appearance in short time by changing the variable value.

* **section1.css and section2.css files:** This file is a **useful and reusable component;** the css is written in such a way that, after including this file in another, we require to set only a few css properties like color, font-size, display etc., to get a visually appealing page.



Above is a portion of code in about.css, a CSS file that imports section2.css. Only a few properties must be set to create section 2 in about.css.

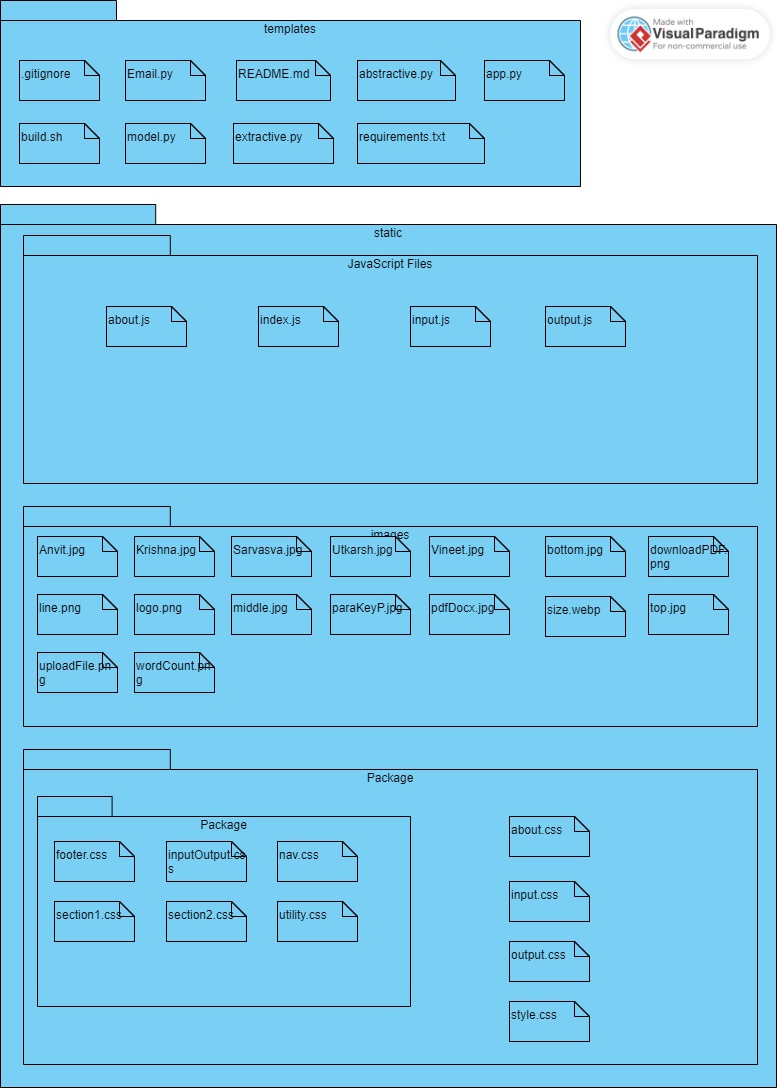
* **nav.css and footer.css:** As the navigation bar and footer are common to all the pages, they serve as **reusable components.**

Dependency Diagram for CSS Files

A diagram of a network

Description automatically generated with medium confidence

Package Diagram



Source Code

The complete source code along with user specifications information is available at:

[Summarix - The Summarizer (github.com)](https://github.com/SarvasvaG/Summarix)

Technologies Used

The technologies used by our "AI Document Summarizer” are:

Frontend: HTML, CSS, JavaScript.

Backend: Python, Flask, SQLite.

Use of the Project in Real Life

The "AI Document Summarizer" project offers a powerful solution to address the information management challenges faced in various real-life scenarios. The practical applications and use cases of the project are wide-ranging and have the potential to significantly impact the way individuals and organizations manage and utilize textual information. Here are some of the key real-life use cases of the project:

**1. Academia and Research:**

* Research Papers and Journals: Researchers and academics can use the AI Document Summarizer to quickly grasp the key findings and contributions of research papers and journals, saving time in the literature review process.
* Thesis and Dissertations: Graduate students can use the tool to efficiently review extensive thesis and dissertation documents, focusing on the main arguments and results.

**2. Journalism and Media:**

* News Articles: Journalists can use the system to rapidly summarize news articles, enabling them to stay updated with breaking news and synthesize information for reporting.
* Press Releases: PR professionals can summarize press releases for quick assessments of important announcements and events.

**3. Legal Practice:**

* Legal Documents: Legal professionals can utilize the system to extract essential information from lengthy legal documents, contracts, and court cases, improving research efficiency.
* Brief Preparation: Preparing concise legal briefs for court proceedings becomes more manageable with the automated summarization of case law and legal precedents.

**4. Business and Corporate Sector:**

* Market Research: Businesses can quickly extract key insights from market research reports, enabling data-driven decision-making.
* Financial Reports: Financial analysts can use the tool to obtain summarized financial reports and corporate disclosures.

**5. Content Creation:**

* Content Strategists: Professionals responsible for content creation and marketing can efficiently extract essential information from articles, blogs, and reports to develop content strategies.
* Content Generation: Automated summarization can aid in the generation of content summaries for websites, blogs, and social media posts.

**6. Education:**

* Educational Materials: Teachers and educators can utilize the system to condense lengthy educational materials, making it easier for students to comprehend and study.
* E-learning: In e-learning platforms, the project can help learners quickly grasp course materials and lectures.

**7. Public Sector:**

* Government Reports: Government officials and policymakers can summarize lengthy government reports, making it easier to understand the implications of policy decisions.
* Public Communications: Summarized versions of public communications can improve accessibility for citizens.

**8. Personal Productivity:**

* Individuals: Anyone dealing with a large volume of text-based content, such as avid readers or individuals conducting research, can use the tool to enhance personal productivity.

**The project's versatility and adaptability make it a valuable asset in various domains and industries. By automating the summarization process, it empowers users to efficiently extract key insights and essential information from extensive documents. This, in turn, streamlines decision-making processes, enhances productivity, and simplifies the challenges posed by the information-rich world in which we live. The "AI Document Summarizer" project serves as a practical and innovative tool with a wide range of real-life applications.**