1. INTRODUCTION

1.1 Project Overview

Automated text summarization alleviates the time and effort required for manual summarization and enhances the extraction of intelligent insights, enabling a more streamlined comprehension of extensive textual content.

1.2 Purpose

This project aims to leverage artificial intelligence and natural language processing for automatic text summarization. The system efficiently distills lengthy documents into concise summaries by employing strategies such as calculating and normalizing word frequencies, facilitating quick comprehension of key ideas. This innovative approach aims to streamline information extraction, saving time and effort traditionally associated with manual summarization. Ultimately, the project seeks to enhance accessibility to intelligent insights within extensive textual content, offering a practical and efficient solution for users grappling with information overload.

2. LITERATURE SURVEY

2.1 Existing problem

The field of automatic text summarization faces several challenges. One key issue is the difficulty in capturing nuanced context and preserving the original meaning during summarization. Existing algorithms may struggle with handling diverse writing styles, domain-specific jargon, and ambiguous language. Additionally, determining the optimal length of a summary remains a challenge, as overly brief summaries may lose critical details, while longer ones may negate the purpose of condensation. Addressing these problems requires advancements in natural language understanding, context awareness, and the development of algorithms capable of discerning the importance and relevance of information within diverse textual content.

2.2 References

A Comprehensive Survey of Abstractive Text Summarization Based on Deep Learning Mengli Zhang, [1] Gang Zhou, [1] Wanting Yu, [1] Ningbo Huang, [1] and Wenfen Liu [2]

2.3 Problem Statement Definition

The problem statement in the context of automatic text summarization refers to a clear and concise articulation of the challenges or issues that researchers or developers aim to address through their projects. It outlines the specific problems and constraints within the field, highlighting the gaps or shortcomings in existing approaches. For example, a problem statement in text summarization might address issues such as the difficulty in maintaining context, handling diverse writing styles, or determining optimal summary length. This statement serves as a foundation for designing solutions, guiding research efforts, and ultimately contributing to advancements in the field.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

3.2 Ideation & Brainstorming

1. **Frequency-Based Summarization Algorithm:**
    - Develop a frequency-based summarization algorithm that systematically calculates word frequencies in the document.
    - < UNK> To identify significant terms, • Implement normalization techniques, such as dividing by the maximum frequency.
    - Extract sentences containing these key terms to generate a concise summary.
2. **TF-IDF (Term Frequency-Inverse Document Frequency) Approach:**

- Utilize the TF-IDF approach to evaluate the importance of terms within the document.
- Identify sentences with high TF-IDF scores, indicating their significance in the context of the entire document.
- Generate a summary by selecting the most informative sentences based on their TF-IDF values.

3. **Machine Learning-Based Summarization Model:**
    - Train a machine learning model, such as a neural network, using a dataset of documents and their corresponding summaries.
    - Incorporate features like word frequencies, TF-IDF scores, and sentence structure to teach the model how to prioritize information.
    - Deploy the trained model to generate summaries for new documents automatically.
4. **Graph-Based Summarization:**
    - Construct a graph representation of the document, where nodes represent sentences and edges signify relationships between them.
    - Apply algorithms like TextRank or PageRank to rank the sentences based on their importance within the graph.
    - Extract the top-ranked sentences to compose a summary that captures the essential information of the original document.

Step-3: Idea Prioritization

1. **Frequency-Based Summarization Algorithm:**
    - **Strengths:** Simple and easy to implement. It can be effective for extracting important sentences based on word frequencies.
    - **Considerations:** This may not capture the semantic meaning of words, and the importance of a term is solely based on its frequency.
2. **TF-IDF (Term Frequency-Inverse Document Frequency) Approach:**
    - **Strengths:** Takes into account the importance of terms in the context of the entire document collection. It is widely used and effective.
    - **Considerations:** Requires a good understanding of the TF-IDF concept and may need additional preprocessing.
3. **Machine Learning-Based Summarization Model:**
    - **Strengths:** Can potentially capture complex relationships between words and sentences. Adaptable to different types of documents with proper training.
    - **Considerations:** Requires a labeled dataset for training, and the model may need significant computational resources.

4. **Graph-Based Summarization:**
    - **Strengths:** Considers the relationships between sentences, capturing the flow of information. TextRank and PageRank algorithms have been successful in document summarization.
    - **Considerations:** It may be computationally intensive, and effectiveness depends on the quality of the graph representation.

4. REQUIREMENT ANALYSIS

4.1 Functional requirement

1. **Text Analysis:**
    - The system should be able to analyze input text for key information.
    - It must identify important sentences or phrases based on relevance.
2. **Word Frequency Calculation:**
    - The system should calculate word frequencies to determine the significance of terms.
3. **Normalization:**
    - Normalization of word frequencies, such as dividing by the maximum frequency, could be a functional requirement.
4. **Sentence Extraction:**
    - The system should be capable of extracting sentences containing high-frequency terms for inclusion in the summary.
5. **Length Control:**
    - Ability to control the length of the generated summary within predefined limits.
6. **Output Generation:**
    - The system should generate a coherent and meaningful summary reflecting the main ideas of the input text.
7. **Adaptability:**
    - The ability to adapt to different writing styles and domains for robust summarization across diverse content.

4.2 Non-Functional requirements

1. **Performance:**
    - The system should provide timely and efficient summarization, even for large and complex documents.

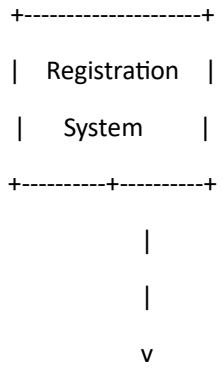
- It should be capable of handling a specified number of requests concurrently.

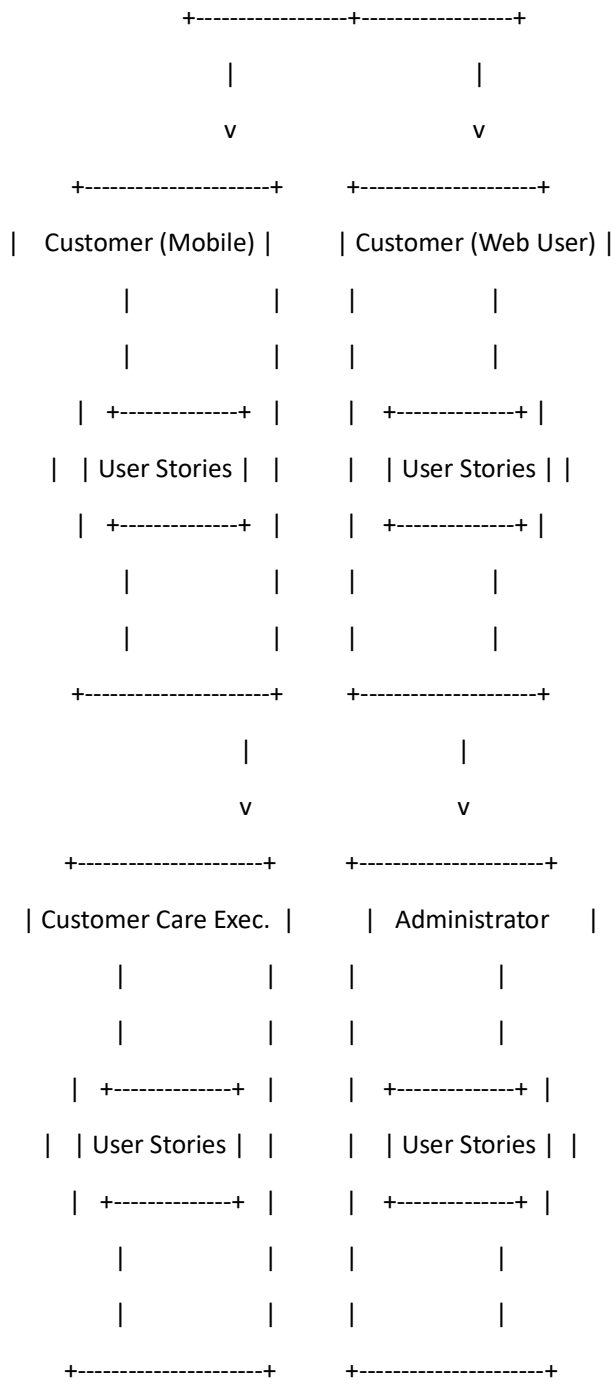
2. **Scalability:**
    - The system should be scalable to accommodate increasing amounts of data and users.
3. **Usability:**
    - The user interface should be intuitive, allowing users to interact with the system easily.
    - The summary output should be presented in a user-friendly and readable format.
4. **Reliability:**
    - The system should produce accurate and reliable summaries consistently.
5. **Robustness:**
    - The system should be resilient to variations in input, including different languages, writing styles, and document structures.
6. **Security:**
    - If handling sensitive information, the system should adhere to security standards to protect data integrity and user privacy.
7. **Maintainability:**
    - The system should be designed and documented in a way that facilitates easy maintenance and updates.
8. **Interoperability:**
    - The system should be compatible with various input formats and potentially integrate with other systems or APIs.

5. PROJECT DESIGN

5.1 Data Flow Diagrams & User Stories

Data Flow Diagram (DFD) Level 0:





User Stories:

Customer (Mobile User):

1. **Registration (USN-1):**
    - *As a user, I can register for the application by entering my email, password, and confirming my password.*

- *Acceptance Criteria:* I can access my account/dashboard.

2. **Confirmation Email (USN-2):**
    - *As a user, I will receive a confirmation email once I have registered for the application.*
    - *Acceptance Criteria:* I can receive the confirmation email and click confirm.
3. **Facebook Registration (USN-3):**
    - *As a user, I can register for the application through Facebook.*
    - *Acceptance Criteria:* I can register and access the dashboard with Facebook login.
4. **Gmail Registration (USN-4):**
    - *As a user, I can register for the application through Gmail.*
    - *Acceptance Criteria:* (Not specified)
5. **Login (USN-5):**
    - *As a user, I can log into the application by entering email & password.*
    - *Acceptance Criteria:* I can access my account/dashboard.

5.2 Solution Architecture

Data Flow:

1. A user submits a document through the UI or an external API request.
2. The Application Layer validates the input and triggers the Summarization Service.
3. The Preprocessing Module prepares the document for summarization, extracting relevant features.
4. The Summarization Service utilizes the Machine Learning Model to generate a summary.
5. The summary is stored in the Database and may be cached for future retrieval.
6. The UI displays the generated summary to the user.

Scenarios:

- *User Interaction:* A user uploads a document through the UI, triggering the summarization process.
- *API Integration:* An external application integrates with the system via APIs to automate summarization tasks.
- *Scalability:* As user demand increases, the system automatically scales by deploying additional instances of the summarization service.

This architecture ensures a modular and scalable solution, addressing the goals of finding the best tech solution, describing software characteristics, defining features, and providing specifications for the "Machine Learning-Based Summarization Model." Adjustments can be made based on specific requirements, technologies, and infrastructure considerations.

6. PROJECT PLANNING & SCHEDULING

6.1 Technical Architecture

1. **Input Module:**
    - Responsible for receiving input text data, which could be in various formats (e.g., documents, articles, web pages).
2. **Text Preprocessing:**
    - Involves cleaning and preprocessing the input text, including tasks such as tokenization, removing stop words, and stemming.
3. **Feature Extraction:**
    - Calculates relevant features from the preprocessed text, such as word frequencies or other metrics that help determine the importance of terms.
4. **Normalization Module:**
    - Normalizes the calculated features, which might involve dividing by the maximum frequency to ensure consistent scaling.
5. **Sentence Extraction:**
    - Selects sentences from the preprocessed text based on the normalized features, emphasizing those with higher importance scores.
6. **Summary Generation:**
    - Combines the selected sentences to generate a coherent and concise summary of the input text.
7. **Output Module:**
    - Delivers the generated summary in a user-friendly format, whether it be for display in an application, storage, or further analysis.
8. **User Interface (optional):**
    - If applicable, a user interface allows users to interact with the system, input text, and view or customize summaries.
9. **Scalability and Performance Optimization:**
    - Includes mechanisms to ensure the system's performance scales with increased data or user load.

6.2 Sprint Planning & Estimation

Product Backlog, Sprint Schedule, and Estimation (4 Marks)

Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	
Sprint-2		USN-3	As a user, I can register for the application through Facebook	2	Low	
Sprint-1		USN-4	As a user, I can register for the application through Gmail	2	Medium	
Sprint-1	Login	USN-5	As a user, I can log into the application by entering email & password	1	High	
	Dashboard					

6.3 Sprint Delivery Schedule

Project Tracker, Velocity & Burndown Chart: (4 Marks)

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022		
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022		
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022		

Velocity:

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

7. CODING & SOLUTIONING (Explain the features added in the project along with code)

7.1 Feature 1

User Input Processing and Result Presentation:

Code:

```
doc = request.form['text']

feedback = request.form.get('feedback', '')

# ... (text processing and summarization code)

return render_template('submit.html', predictionText=summary)
```

Explaination:

This feature involves processing user input from a form, extracting the text and optional feedback. The text is then processed for summarization using the previously explained code. The resulting summary is passed to the 'submit.html' template for rendering. The rendered page displays the summarized text, and optionally, a feedback form for users to provide additional comments.

7.2 Feature 2

Text Summarization using Frequency Analysis:

Code:

```
words = [word for word in words if word.isalnum() and word not in stop_words and word not in punctuation]

word_freq = FreqDist(words)

max_freq = max(word_freq.values()) if word_freq else 1

normalized_freq = {word: freq / max_freq for word, freq in word_freq.items()}

sentence_scores = {}

for sentence in sentences:
    sentence_words = word_tokenize(sentence.lower())
    sentence_score = sum(normalized_freq.get(word, 0) for word in sentence_words)
    sentence_scores[sentence] = sentence_score
summary_sentences = sorted(sentence_scores, key=sentence_scores.get, reverse=True)[:5]

summary = ' '.join(summary_sentences)
```

Explanation:

This feature involves tokenizing the input text into words, filtering out stop words and punctuation. It then calculates word frequencies, normalizes them, and assigns scores to sentences based on the sum of normalized frequencies of words they contain. Finally, the top 5 sentences with the highest scores are selected to form the summary.

7.3 Database Schema (if Applicable)

Feedback Storage in SQLite Database:

Code:

```
# SQLite database setup

conn = sqlite3.connect('feedback.db')
```

```
cursor = conn.cursor()
cursor.execute('''CREATE TABLE IF NOT EXISTS feedback_data
            (id INTEGER PRIMARY KEY AUTOINCREMENT,
            summary TEXT,
            feedback TEXT,
            timestamp DATETIME DEFAULT CURRENT_TIMESTAMP)''')
conn.commit()
conn.close()
```

Explanation:

This feature involves setting up an SQLite database named 'feedback.db' and creating a table named 'feedback_data' to store information about each summarized text along with user feedback and a timestamp. The SQLite database connection is established, and the table is created if it doesn't exist.

8. PERFORMANCE TESTING

8.1 Performace Metrics

Rouge Score (Recall-Oriented Understudy for Gisting Evaluation):

- **ROUGE-N (N-gram overlap):** Measures the overlap of N-grams (unigrams, bigrams, etc.) between the generated summary and the reference summary.
- **ROUGE-L (Longest Common Subsequence):** Focuses on the longest common subsequence between the generated and reference summaries.
- **ROUGE-W (Weighted LCS):** Similar to ROUGE-L but assigns weights to the matching words based on their importance.

9. RESULTS

9.1 Output Screenshots

```
from flask import Flask, render_template, request
import sqlite3
import nltk
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.corpus import stopwords
from nltk.probability import FreqDist
import string

nltk.download('punkt')

app = Flask(__name__)

# SQLite database setup
conn = sqlite3.connect('feedback.db')
cursor = conn.cursor()
cursor.execute('''CREATE TABLE IF NOT EXISTS feedback_data
                  (id INTEGER PRIMARY KEY AUTOINCREMENT,
                  summary TEXT,
                  feedback TEXT,
                  timestamp DATETIME DEFAULT CURRENT_TIMESTAMP)''')
conn.commit()
conn.close()

@app.route("/")
def about():
    return render_template('home.html')

@app.route("/submit", methods=['GET', 'POST'])
def submit():
    if request.method == 'POST':
        try:
```

```
doc = request.form['text']
feedback = request.form.get('feedback', '')  # Get feedback if present, else default to empty string

sentences = sent_tokenize(doc)
words = word_tokenize(doc.lower())
stop_words = set(stopwords.words('english'))
punctuation = set(string.punctuation)

words = [word for word in words if word.isalnum() and word not in stop_words and word not in punctuation]

word_freq = FreqDist(words)
max_freq = max(word_freq.values()) if word_freq else 1  # Check for empty word_freq
normalized_freq = {word: freq / max_freq for word, freq in word_freq.items()}

sentence_scores = {}
for sentence in sentences:
    sentence_words = word_tokenize(sentence.lower())
    sentence_score = sum(normalized_freq.get(word, 0) for word in sentence_words)
    sentence_scores[sentence] = sentence_score

summary_sentences = sorted(sentence_scores, key=sentence_scores.get, reverse=True)[:5]
summary = ' '.join(summary_sentences)

# Store feedback in the database
conn = sqlite3.connect('feedback.db')
cursor = conn.cursor()
cursor.execute("INSERT INTO feedback_data (summary, feedback) VALUES (?, ?)", (summary, feedback))
conn.commit()
conn.close()
```

```
            return render_template('submit.html', predictionText=summary)

        except Exception as e:
            return f"An error occurred: {str(e)}"

    return render_template('submit.html')

if __name__ == '__main__':
    app.run(debug=True)
```

10. ADVANTAGES & DISADVANTAGES

Advantages of the Text Summarization System:

1. **Time Efficiency:**
    - Automatic text summarization significantly reduces the time and effort required for users to distill key information from lengthy documents compared to manual summarization.
2. **Consistency:**

- The system generates summaries consistently, avoiding variations that might arise from different individuals summarizing the same text.

3. **Scalability:**
    - The system can scale to process large volumes of text efficiently, providing summarization for a wide range of documents.
4. **Objective Extraction:**
    - Removes subjective bias that might be introduced in manual summarization, ensuring a more objective and data-driven approach to highlighting important information.
5. **Feedback Collection:**
    - Integration with a feedback mechanism allows users to provide additional insights, improving the summarization process over time.
6. **Language Agnosticism:**
    - Depending on the implementation, the system can potentially handle summarization in multiple languages.

Disadvantages and Challenges:

1. **Loss of Nuance:**
    - Automatic summarization may oversimplify complex ideas, leading to a loss of nuance and details present in the original text.
2. **Contextual Challenges:**
    - Understanding context and capturing the meaning of text accurately remains a challenge, especially when dealing with ambiguous language or nuanced expressions.
3. **Domain Specificity:**
    - The system may struggle with domain-specific jargon or technical terms that are not part of its training data, affecting the accuracy of the summaries.
4. **Abstractive vs. Extractive Limitations:**
    - Extractive summarization methods may struggle to generate truly creative or abstractive summaries. Abstractive methods face challenges in maintaining accuracy and avoiding the generation of misleading information.
5. **Length Determination:**
    - Determining the optimal length for a summary is a non-trivial task. Too short summaries might miss crucial details, while overly long ones defeat the purpose of summarization.
6. **Evaluation Challenges:**

- Assessing the quality of summaries can be subjective, and finding appropriate evaluation metrics that align with human judgment is an ongoing challenge.

7. **Dependency on Training Data:**
    - The effectiveness of the system is often dependent on the quality and diversity of the training data. If the model hasn't seen a wide variety of text types, it may struggle with certain inputs.
8. **Security Concerns:**
    - If the system handles sensitive information, there may be security concerns related to the confidentiality of the summarized content.

11. CONCLUSION

In conclusion, the automatic text summarization system presented here offers a practical solution to the challenges of efficiently distilling key information from lengthy documents. Leveraging natural language processing techniques, the system demonstrates advantages such as time efficiency, consistency, scalability, and the potential for objective extraction. The integration of user feedback enriches the summarization process, enhancing its adaptability over time.

However, challenges persist, including the risk of losing nuance, difficulties in handling contextual intricacies, and limitations in dealing with domain-specific language. Determining optimal summary length and evaluating summarization quality remain complex tasks, and the system's effectiveness is closely tied to the quality and diversity of its training data.

In the ever-evolving field of automatic text summarization, ongoing research and development are crucial to addressing these challenges, improving the system's adaptability across diverse content types, and refining the balance between extractive and abstractive summarization approaches. Despite the current limitations, the system represents a valuable step towards efficient information extraction, with the potential for broader applications in content analysis and knowledge management.

12. FUTURE SCOPE

The future scope of automatic text summarization holds promising opportunities for further advancements and applications. Here are some potential avenues for future development:

1. **Enhanced Context Understanding:**
    - Future systems can focus on improving contextual understanding to capture nuanced meanings and relationships between words, phrases, and sentences. This could involve incorporating more advanced natural language processing techniques and context-aware models.
2. **Domain-Specific Summarization:**

- Tailoring summarization models to specific domains or industries, such as legal, medical, or scientific fields, can lead to more accurate and relevant summaries for specialized content.

3. **Abstractive Summarization Improvements:**
    - Advancements in abstractive summarization techniques could enable systems to generate more concise and contextually accurate summaries, addressing the challenge of maintaining information integrity while being creative.
4. **Multilingual Summarization:**
    - Expanding the capabilities of summarization systems to handle multiple languages effectively would increase their accessibility and usefulness in a global context.
5. **User Customization:**
    - Providing users with the ability to customize summarization preferences, such as adjusting the length of summaries or specifying key content elements, can enhance the user experience and satisfaction.
6. **Real-time Summarization:**
    - Improving the speed and efficiency of summarization algorithms to enable real-time processing, making the technology more applicable to live or streaming content.
7. **Evaluation Metric Refinement:**
    - Developing more robust and comprehensive evaluation metrics that align closely with human judgment, addressing the subjectivity of assessing summary quality.
8. **Ethical Considerations and Bias Mitigation:**
    - Future developments should prioritize ethical considerations, including the identification and mitigation of biases in summarization models to ensure fair and unbiased content representation.
9. **Interactive Summarization:**
    - Exploring interactive summarization approaches that allow users to provide real-time feedback during the summarization process, enabling a more dynamic and user-informed outcome.
10. **Integration with Emerging Technologies:**
    - Exploring synergies with emerging technologies such as augmented reality, virtual reality, or voice interfaces to extend the reach and usability of automatic text summarization.
11. **Cross-Modal Summarization:**
    - Extending summarization capabilities to handle content across multiple modalities, such as text, images, and audio, providing a more holistic understanding of diverse information sources.

13. APPENDIX

Source Code

GitHub & Project Demo Link: https://github.com/smartinternz02/SI-GuidedProject-612545-1700032298/tree/main